TESIS - KS142501

PERBANDINGAN STEMMER BAHASA INDONESIA DAN DAMPAKNYA PADA PENGGALIAN TEKS BAHASA INDONESIA, STUDI KASUS PENGELOMPOKAN KELUHAN PELANGGAN PLN

AFIAN SYAFAADI RIZKI
5214201201

DOSEN PEMBIMBING
Dr. Ir. Aris Tjahyanto, M.Kom.

PROGRAM MAGISTER
JURUSAN SISTEMINFORMASI
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Halaman ini sengaja dikosongkan

TESIS- KS142501

Perbandingan Stemmer Bahasa Indonesia dan Dampaknya pada Penggalian Teks Bahasa Indonesia, Studi Kasus Pengelompokan Keluhan Pelanggan PLN

AFIAN SYAFAADI RIZKI
NRP.521 420 1201

DOSEN PEMBIMBING
Dr. Ir. ARIS TJAHYANTO, M.Kom
NIP. 196503101991021001

PROGRAM MAGISTER
JURUSAN SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Halaman ini sengaja dikosongkan

THESIS- KS142501

Comparison of Stemming Methods and Its Impact on Indonesian Text Processing, A Case Study of PLN's Customer's Complaints

AFIAN SYAFAADI RIZKI
NRP.521 420 1201

SUPERVISOR
Dr. Ir. ARIS TJAHYANTO, M.Kom
NIP. 196503101991021001

MASTER PROGRAM
DEPARTMENT OF INFORMATION SYSTEMS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M. Kom)
di
Institut Teknologi Sepuluh Nopember

oleh:
Afian Syafaadi Rizki
NRP 5214201201

Tanggal Ujian : 11 Juli 2017
Periode Wisuda: September 2017

Disetujui oleh:

1. Dr. Ir. Aris Tjahyanto., M.Kom **(Pembimbing)**
NIP. 19650310 199102 1001

2. Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom **(Penguji)**
NIP. 19730219 199802 1001

3. Faizal Mahananto, S.Kom., M.Eng, Ph.D **(Penguji)**
NIK. 5200.201301010

Dekan Fakultas Teknologi Informasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom
NIP 19720809 199512 1001

Halaman ini sengaja dikosongkan

Perbandingan Stemmer Bahasa Indonesia dan Dampaknya pada Penggalian Teks Bahasa Indonesia, Studi Kasus Pengelompokan Keluhan Pelanggan PLN

Nama Mahasiswa : Afian Syafaadi Rizki
NRP : 5214201201
Pembimbing : Dr. Ir. Aris Tjahyanto, M.Kom

ABSTRAK

Stemming merupakan salah satu tahap yang dilakukan pada proses penggalian informasi dari teks. Proses *stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. Metode *stemmer* dapat dievaluasi dengan dua cara, yang pertama adalah mengukur seberapa akurat metode *stemmer* untuk mengubah kata menjadi kata dasar. Cara yang kedua adalah mengukur seberapa besar pengaruh yang diberikan metode *stemmer* pada performa penggalian informasi, misalnya berapa persentase kenaikan hasil klasifikasi dokumen dengan menerapkan metode *stemmer*. Pada penelitian ini akan dibandingkan metode-metode *stemmer* Bahasa Indonesia dan satu metode *stemmer* Bahasa Inggris. Metode *stemmer* yang digunakan yaitu *porter confix striping stemmer, nazief stemmer, arifin stemmer, fadillah stemmer, asian stemmer, enhanced confix stripping stemmer, dan arifiyanti stemmer.* Metode pengelompokan data yang digunakan adalah *k-means* sedangkan untuk mengevaluasi hasil pengelompokan data/ *cluster,* digunakan *davies bouldin index* (DBI). Untuk mengukur performa metode-metode *stemmer* dalam mengubah kata menjadi kata dasar digunakan metode evaluasi *paice* yang meliputi perhitungan indeks *understemming* dan *overstemming.* Selain itu, digunakan pengujian *t* terhadap rata-rata nilai DBI untuk mengetahui signifikansi dari peningkatan performa pengelompokan data. Hasil penelitian menunjukkan bahwa metode *stemmer* yang paling akurat adalah *enhached confix stripping stemmer* dengan nilai *understemming* 0,06845 dan nilai *overstemming* 0,00116, sedangkan performa terendah adalah *porter confix striping stemmer* dengan nilai *understemming* 0,22661 dan nilai *overstemming* 0,0. Untuk metode *stemmer* yang mampu meningkatkan performa *clustering* secara signifikan adalah metode *arifiyanti stemmer* dengan rata-rata DBI 8,673 dan *p-value* 0.01496, metode *asian stemmer* dengan rata-rata DBI 8,696 dan *p-value* 0.00018, dan metode *nazief stemmer* dengan rata-rata DBI 8,759, dan *p-value* 0.01405.

Kata Kunci : *Stemming* Bahasa Indonesia, *text mining, information retrieval, Confix Stripping Stemmer, k-means, paice evaluation*

Halaman ini sengaja dikosongkan

Comparison of Stemming Methods and Its Impact on Indonesian Text Processing, A Case Study of PLN's Customer's Complaints

ABSTRACT

Stemming is one of the stages performed for extracting information from a text. The *stemming* process is a process of converting a word into its root. The stemmer method can be evaluated in two ways, the first is to measure how accurate the method is in converting a word into its root. The second way is to measure how much influence the stemmer method has in improving the performance of information retrieval, for example, how much is the improvement of document classification result by applying stemmer method. In this study the researcher compared the stemming methods of Indonesian language and an English stemmer method. The stemmer methods used were *porter confix stripping stemmer, nazief stemmer, arifin stemmer, fadillah stemmer, asian stemmer, enhanced confix stripping stemmer*, and *arifiyanti stemmer*. The method used for clustering the data was *k-means* while to evaluate the result of cluster, this study used *davies bouldin index* (DBI). To measure the performance of stemmer methods in converting words into roots, *paice* evaluation method was used, which included calculation of *understemming* and *overstemming* index. In addition, a *t*-test against the average values of DBI was performed to find out the significance of the improved performance in data clustering. The results show that the most accurate stemmer method was *enhanced confix stripping stemmer* with *understemming* value of 0.06845 and *overstemming* value of 0.00116, while the lowest performance was*porter confix stripping stemmer* with *understemming* value of 0.22661 and *overstemming* value of 0.0. The stemming methods which can significantly improve data clustering performance were*arifiyanti stemmer* with the average of DBI 8.673 and *p-value*of 0.01496, asian stemmer with the average of DBI 8.696 and *p-value*of 0.00018, and *nazief stemmer* with the average of DBI 8.759, and *p-value*of 0.01405.

Keywords : *Stemming* Indonesian language, *text mining, information retrieval, confix stripping stemmer, k-means, paice evaluation*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala rahmat dan ridho-Nya, sehingga penulis dapat menyelesaikan tesis dengan judul "Perbandingan Stemmer Bahasa Indonesia dan Dampaknya pada Penggalian Teks Bahasa Indonesia, Studi Kasus Pengelompokan Keluhan Pelanggan PLN". Tesis ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan pada Program Magister Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember. Penyusunan tesis ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada :

1. Bapak Dr. Ir. Aris Tjahyanto, M.Kom selaku dosen pembimbing yang telah meluangkan waktu, tenaga, dan pikiran, serta memberikan ilmu, pengalaman, arahan, perhatian, dukungan, dan kesabaran selama membimbing penulis dari awal menempuh pendidikan magister hingga tesis ini selesai.
2. Bapak Dr. Eng. Febriliyan Samopa, M.Kom selaku Dosen Penguji I dan bapak Faizal Mahananto, S.Kom., M.Eng, Ph.D selaku Dosen Penguji II, yang telah bersedia menguji, memberikan masukan, kritik, dan saran untuk perbaikan tesis ini.
3. Orang tua penulis, bapak Ir. Maheru Anindito. Beserta orang tua wali penulis, bapak Drs. Fachrurrazy, M.A., Ph.D, dan ibu Dra. Sri Widati, yang senantiasa mendukung, memberikan do'a, memberikan pengarahan, dan memberikan motivasi kepada penulis.
4. Seluruh Bapak dan Ibu dosen program magister jurusan Sistem Informasi ITS, yang telah membagi banyak ilmu dan inspirasi kepada penulis.
5. Rekan-rekan kuliah program magister jurusan Sistem Informasi ITS angkatan 2014 dan 2015, yang memberi bantuan dan dukungan kepada penulis dalam menyelesaikan studi.
6. Teman-teman lama penulis yang berdomisili di Surabaya, yang memberikan dukungan dan menemani penulis menjelajahi kota Surabaya.

7. Semua pihak terkait yang tidak dapat disebutkan satu-persatu. Yang selama pengerjaan penelitian ini telah memberikan dukungan, bantuan, dan jasanya terhadap penulis.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan tesis ini, karena penulis menyadari banyaknya kekurangan. Akhirnya penulis berharap, semoga tesis ini dapat bermanfat.

Surabaya, Juli 2017

Afian Syafaadi Rizki

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR LAMPIRAN

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Tesis ini membahas tentang perbandingan metode *stemmer* bahasa Indonesia dan dampaknya pada penggalian teks bahasa Indonesia, sebuah studi kasus tentang pengelompokan keluhan pelanggan PLN. Pada Bab Pendahuluan ini dibahas tentang latar belakang penelitian, rumusan masalah, tujuan penelitian, kontribusi penelitian, batasan penelitian, dan sistimatika penulisan tesis.

1.1 Latar Belakang

Penggalian teks atau disebut *text mining* adalah proses penggalian informasi dari database berukuran besar untuk menemukan pola-pola yang tidak diketahui (Bhanuse dkk, 2016). Penggalian teks melibatkan seperangkat teknikuntuk mengelompokkan, mengklasifikasikan dan mengekstrak informasi yang relevan dari sekumpulan data software requirements. Teknik *text mining* merupakan bagian dari teknik *Knowledge Discovery from Database* (KDD), yang merupakan proses semi otomatis untuk melakukan penggalian pengetahuan yang relevan dari data base,dan bertujuan untuk menemukan informasi yang sebelumnya tidak diketahui, serta berpotensi bermanfaat.

Penerapan penggalian teks telah diaplikasikan untuk menyelesaikan permasalahan pada berbagai bidang, diantaranya untuk peramalan (Seo dkk, 2016), ekonomi (Wang dkk, 2016), kesehatan (Ali dkk, 2016; Urbain, 2015), pengembangan perangkat lunak (Bakar dkk, 2015; Achimugu dkk, 2014), politik (Charalampakisdkk, 2016), analisis sentimen (Fattah, 2015), analisis emosi (PerikosdanHatzilygeroudis, 2016). Tantangan yang dihadapi dalam pemrosesan teks diantaranya sangat beragamnya bahasa yang digunakan, besarnya dimensi data, dan besarnya *data noise*. Sebagian besar penelitian yang dilakukan berasumsi bahwa data teks yang digunakan telah benar secara struktur bahasa, sedangkan beberapa peneliti yang lain memperhatikan struktur bahasa dalam penelitian mereka (Petz dkk, 2014).Untuk mengurangi *data noise*, perlu dilakukan pra-proses data. Terdapat beberapa metode yang dapat digunakan dalam tahap pra-proses, antara lain *stopping* atau menghapus *stopwords*,

mengubah karakter *emoticon* menjadi karakter teks, dan *stemming* (Sari dkk, 2014).

Proses *stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. *Stemming* dilakukan sebagai cara untuk meningkatkan performa penggalian teks karena kata yang memiliki kata dasar yang sama dianggap memiliki kemiripan makna. Dokumen-dokumen yang memiliki kata-kata dengan kata dasar yang sama dianggap relevan sehingga proses *stemming* akan mengurangi dimensi fitur (Rajput dan Khare, 2015).

Algoritma *stemmer* terbagi menjadi dua jenis yaitu *stemmer* berbasis statistik (*statistic-based stemmer*) dan berbasis aturan (*rule-based stemmer*). *Statistic-based stemmer* merupakan metode *supervised* yang menggunakan data latih untuk membentuk model berupa parameter dan aturan dalam melakukan *stemming*, sedangkan *rule-based stemmer* merupakan metode *unsupervised* yang menggunakan sejumlah aturan yang sudah didefinisikan untuk melakukan *stemming*. Keunggulan dari *statistic-based stemmer* adalah dapat diterapkan pada bahasa yang berbeda dengan lebih mudah, memungkinkan untuk melakukan *stemming* multi bahasa. Kekurangan dari *statistic-based stemmer* adalah membutuhkan data dengan jumlah yang cukup besar untuk mendapat akurasi yang bagus sehingga kurang cocok untuk jumlah data sedikit, tidak dapat menangani kata yang diluar model statistik, contoh "sing-song", "foot-feet" (Brychcíndan Konopík, 2015). Keunggulan dari *rule-based stemmer* adalah memiliki akurasi yang tinggi, tidak membutuhkan data latih (*unsupervised*). Kekurangan dari *rule-based stemmer* adalah hanya dapat digunakan untuk satu bahasa, karena melakukan *stemming* untuk bahasa lain membutuhkan definisi aturan yang berbeda, oleh sebab itu sangat sulit untuk melakukan *stemming* multi bahasa.

Pada penelitian yang dilakukan oleh Brychcín dan Konopík (2015), diusulkan metode *stemmer* berbasis statistik yang disebut High Precision *Stemmer* (HPS). Algoritma *stemmer* tersebut terdiri dari dua tahap, yang pertama tahap pengelompokan kata-kata yang memiliki kesamaan konteks. Kelompok-kelompok kata tersebut kemudian dibuat kata dasarnya. Tahap kedua, prosesnya adalah klasifikasi kata yang akan dicari kata dasarnya akan diklasifikasikan ke kelompok-kelompok kata yang sudah terbentuk pada tahap pertama. Karena

metode tersebut membutuhkan kelompok-kelompok kata sebagai data latih, maka metode tersebut termasuk *supervised.* Beberapa metode *stemmer* yang dibandingkan adalah *graph- based stemmer* (GRAS), YAAS, Linguistica, dan Rule-Based *Stemmer*. Bahasa yang digunakan adalah bahasa Czech, bahasa Slovakia, bahasa Poland, bahasa Hungaria, bahasa Spanyol, dan bahasa Inggris. Khusus untuk Polandia dan Slovakia tidak ditemukan metode rule-based *stemmer*. Hasil penelitian menunjukkan bahwa HPS *stemmer* merupakan yang paling efektif dan paling *universal* untuk melakukan *stemming* terhadap bahasa yang memiliki *morphology* yang kaya. GRAS sangat efisien untuk kasus *information retrieval* dan memiliki performa yang bagus untuk *inflectional removal.* YAAS memberikan hasil yang konsisten dan juga sangat *universal.* Untuk mengetahui pengaruh *stemming* terhadap *information retrieval*, peneliti menggunakan mean average precision (MAP) dan menggunakan uji statistik *t* dengan *confidence level 0.95*, hasilnya menunjukkan bahwa metode HPS memberikan hasil yang baik walaupun metode tersebut tidak dirancang khusus untuk kasus *information retrieval.* Kesimpulan lain yang didapatkan adalah metode *stemmer* yang diusulkan memberikan peningkatan nilai *F-measure* mulai jumlah 50.000 kata sampai 5.000.000 kata(Brychcín dan Konopík, 2015).

Pada penelitian oleh Flores dan Moreira(2016), dilakukan perbandingan metode-metode *stemming* untuk empat bahasa yaitu bahasa Inggris, bahasa Perancis, bahasa Portugis, dan bahasa Spanyol. Metode-metode *stemming* yang digunakan yaitu Lovins, Paice/Husk, Porter, UEA-Lite, Linguistica, *Stemmer*-S, GRAS, dan Trunc3 sampai Trunc8. Peneliti melakukan perbandingan performa setiap algoritma *stemmer* dan dibandingkan juga dengan tanpa *stemmer* (*NoStem*). Metode evaluasi untuk *stemmer* menggunakan *Error Rate Relative to Trunction* (ERRT), sedangkan untuk mengevaluasi dampak pada *Information retrieval* digunakan *average precision* (AvP) dan *mean average precision* (MAP). Peneliti juga melakukan uji statistik *t* dengan α=0,05 untuk melihat apakah ada pengaruh signifikan dari penerapan setiap metode *stemmer.*Hasil pengujian menunjukkan bahwa untuk bahasa Portugis, *stemmer* yang signifikan adalah UniNE, Porter, *Stemmer*-S, Trunc5, dan Trunc7. Untuk bahasa Perancis, *stemmer* yang signifikan adalah UniNE, Porter, RSLP Paice/Husk, GRAS, Trunc5, Trunc6, Trunc7, dan

Trunc8. Untuk bahasa Spanyol, *stemmer* yang signifikan adalah UniNE, Porter, GRAS, Trunc5, Trunc6, Trunc7, dan Trunc8. Untuk bahasa Inggris, *stemmer* yang signifikan adalah *Stemmer*-S dan GRAS. Untuk bahasa Inggris, hasil MAP Trunc7 adalah 0,329 dan *Stemmer*-S 0,324, sedangkan MAP tertinggi didapatkan dengan *stemmer* Trunc6 yaitu 0,331. Untuk bahasa Portugis, hasil MAP Trunc7 adalah 0,281 dan *Stemmer*-S 0,273 , sedangkan MAP tertinggi didapatkan dengan *stemmer* UniNE yaitu 0,298. Untuk bahasa Perancis, hasil MAP Trunc7 adalah 0,266 dan *Stemmer*-S 0,268 , sedangkan MAP tertinggi didapatkan dengan *stemmer* UniNE yaitu 0,273. Untuk Bahasa Spanyol, hasil MAP Trunc7 adalah 0,303 dan *Stemmer*-S 0,298 , sedangkan MAP tertinggi didapatkan dengan *stemmer* UniNE yaitu 0,317(Flores dan Moreira, 2016).

Kesimpulan dari penelitian tersebut menunjukkan bahwa algoritma *stemmer* yang paling akurat bukan satu-satunya cara untuk mendapatkan peningkatan pada *information retrieval* di semua Bahasa, hasil eksperimen juga menunjukkan bahwa *low stemmer* (Trunc7, *Stemmer*-S) walaupun sederhana tetapi dapat memberikan peningkatan pada semua bahasa. Penelitian selanjutnya dapat dilakukan untuk mengetahui apakah korelasi yang ditemukan akan berlaku untuk bahasa lain yang memiliki karakteristik berbeda (Flores dan Moreira, 2016).

Untuk bahasa Indonesia, terdapat beberapa algoritma *stemmer* yang sering digunakan yaitu algoritma Nazief, algoritma Vega, algoritma Asian, algoritma Arifin, dan algoritma Fadillah. Dari beberapa algoritma tersebut, seluruhnya berdasar pada *confix stripping stemmer,* yaitu menggunakan pemotongan imbuhan kata berdasarkan ejaan bahasa Indonesia, sehingga algoritma-algoritma tersebut tergolong *rule-based stemmer*.

Proses *stemming* metode-metode tersebut belum maksimal, sehingga dilakukan beberapa perbaikan pada metode Asian (2007). Asian(2007) telah membuktikan bahwa algoritma *confix stripping* memiliki hasil yang paling baik, walaupun tetap masih ada kekurangan. (Arifin dkk, 2009) melakukan perbaikan aturan untuk*prefix* mem+p, men+s, menge+…, penge+…, sedangkan Arifiyanti (2015) menambahkan aturan untuk bahasa tidak baku. Kekurangan pada penelitian tersebut diantaranya pada tahap normalisasi karakter berulang, penghapusan terjadi pada kata yang memang memiliki huruf ganda, misalnya

"saat" menjadi "sat", hal ini menyebabkan kata tidak dapat diproses dengan baik oleh *stemmer*(Arifiyanti2015).

Perbandingan metode-metode *stemmer* untuk bahasa Indonesia dapat dilihat pada Tabel 1.1. Adapun algoritma yang juga ikut dibandingkan adalah *Porter confix stripping stemmer* yang menjadi dasar sebagian besar metode metode-metode *stemmer* untuk bahasa Indonesia.

Tabel 1.1 Perbandingan Metode *Stemmer* Bahasa Indonesia

Nomor	**Nama Metode**	**Keterangan**	**Sumber**
1	*Porter confix stripping stemmer*	• Metode *Stemmer*untuk Bahasa Inggris • Merupakan *Rule-based Stemmer* • Menggunakan sejumlah aturan yang dibagi menjadi lima langkah. • Menggunakan huruf *vowel* (vokal) yang diikuti konsonan sebagai *measure condition*	*An algorithm for suffix stripping. Porter.* (Porter, 1980)
2	*Nazief Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer* • Menggunakan sejumlah aturan yang dibagi menjadi enam langkah. • Menggunakan kamus kata dasar • *Menggunakan daftar awalan-akhiran yang tidak diperbolehkan*	Confix Stripping : Approach to Stemming Algorithm forBahasa Indonesia . *Internal Publication.* Faculty of Computer Science, University of Indonesia. (Nazief,1996).
3	*Arifin Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer* • Setiap kata diasumsikan memiliki 2 awalan (prefix) dan 3 akhiran (sufiks).	Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering.

Tabel 1.1 Perbandingan Metode *Stemmer* Bahasa Indonesia (Lanjutan)

Nomor	Nama Metode	Keterangan	Sumber
		(lanjutan)	(Arifin dan Setiono, 2002)
4	*Fadillah Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer* • Aturan disusun berurutan. Apabila aturan pertama gagal, maka aturan selanjutnya dijalankan, dan seterusnya. • Tidak menggunakan kamus kata dasar • Menggunakan suku kata sebagai *measure condition*	A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. (Tala, 2003)
5	*Asian Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer* • Merupakan Perbaikan dari *Nazief Stemmer* • *Perbaikan dilakukan dengan menambah aturan dan dilakukan penanganan untuk bentuk jamak.* • *Menggunakan kamus kata dasar*	Stemming Indonesian (Asian, 2007)
6	*Enhanced Confix Stripping Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer* • Merupakan Perbaikan dari *Nazief Stemmer dan Asian Stemmer.* • Perbaikan dilakukan dengan menambah 5 aturan penghapusan awalan. • Menggunakan kamus kata dasar	*Enhanced Confix Stripping Stemmer and Ants Algorithm for Classifying News Documents in Indonesian Language.* (Arifin dkk, 2009)
7	*Arifiyanti Stemmer*	• Metode *Stemmer*untuk Bahasa Indonesia • Merupakan *Rule-based Stemmer*	(Arifiyanti, 2015)

Dari penjabaran diatas, penulis melakukan perbandingan metode-metode *stemmer* bahasa Indonesia dengan tujuan mengetahui metode-metode yang meningkatkan *information retrieval* secara signifikan. Selain itu juga untuk mengetahui apakah korelasi yang ditemukan oleh Flores dan Moreira(2016) akan berlaku pada bahasa Indonesia. Untuk itu perlu dilakukan pengujian statistik *t* untuk setiap metode *stemmer*. Twitter PLN dipilih sebagai studi kasus, diantaranya karena adanya pelanggan yang mengeluhkan keluhan yang serupa berulangkali, dan ada pelanggan yang tidak mendapat respon dari pihak perusahaan.

Penulis memilih metode pengelompokan (*clustering*) dibandingkan metode klasifikasi. Pemilihan tersebut atas pertimbangan tidak adanya dasar pembagian kelas keluhan pelanggan PLN, selain itu penggunaan metode klasifikasi akan membatasi kemungkinan adanya kelas baru. Adapun metode yang dipilih adalah *k-means* karena metode tersebut memiliki ketelitian cukup tinggi terhadap ukuran objek, relatif lebih terukurdan efisien untuk pengolahan objek dalam jumlah besar. (Ediyanto dkk, 2013).

1.2 Rumusan Masalah

Berdasarkan penjabaran dalam latar belakang, maka masalah yang mendasari penelitian ini dirumuskan sebagai berikut:

1) Bagaimana melakukan penerapan metode-metode *stemmer* bahasa Indonesia untuk data keluhan pelanggan PLN.

2) Bagaimana perbandingan metode-metode *stemmer* terhadap i*nformation retrieval* pada bahasa Indonesia.

3) Bagaimana hubungan antara akurasi metode *stemmer* (Indeks *Under Stemming*, Indeks *Over Stemming*) terhadap performa *information retrieval*pada bahasa Indonesia.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dijabarkan, tujuan dari penelitian ini adalah:

1) Menerapkan metode-metode *stemmer* bahasa Indonesia pada data keluhan pelanggan PLN.
2) Mendapatkan metode-metode *stemmer* yang dapat meningkatkan performa *information retrieval* bahasa Indonesia.
3) Mengkonfirmasi apakah temuan Flores dan Moreira(2016) akan berlaku pada bahasa Indonesia. Temuannya yaitu algoritma *stemmer* yang paling akurat bukan satu-satunya cara untuk mendapatkan peningkatan di *information retrieval*, dan *low stemmer* walaupun sederhana tetapi dapat memberikan peningkatan pada *information retrieval*akan berlaku pada bahasa Indonesia.

1.4 Kontribusi Penelitian

Hasil penelitian ini diharapkan dapat memberikan kontribusi baik secara teori maupun secara praktik. Kontribusi secara teori didapat dari hasil perbandingan metode *stemmer* pada tahap pra-proses penggalian teks. Hasil perbandingan metode tersebut dapat digunakan sebagai masukan untuk berbagai jenis penelitian penggalian teks yang akan datang.

Sedangkan kontribusi secara praktik bagi PLN didapatkan dari hasil pengelompokan keluhan. Informasi dari teks keluhan pelanggan yang didapatkan akan dijadikan dasar untuk membantu penanganan keluhan pelanggan PLN.

1.5 Batasan Penelitian

Batasan pada penelitian ini diantaranya adalah penggunaan data menggunakan media sosial *twitter*. Selain itu, data hanya diambil dari akun resmi PLN yaitu @pln_123 pada periode Agustus sampai September 2016.

1.6 Sistematika Penulisan Tesis

Sistematika penulisan dokumen laporan penelitian tesis ini dibagi menjadi lima bab yakni sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian, kontribusi penelitian, batasan penelitian, dan sistematika penulisan.

BAB 2KAJIAN PUSTAKADANLANDASAN TEORI

Pada bab ini dijelaskan mengenai kajian pustaka dari berbagai penelitian yang memiliki kaitan dengan penelitian ini. Kajian pustaka ini bertujuan untuk memperkuat dasar dan alasan dilakukannya penelitian ini. Selain kajian pustaka, pada bab ini juga dijelaskan mengenai teori-teori terkait yang bersumber dari buku, jurnal, ataupun artikel yang berfungsi sebagai dasar dalam melakukan penelitian agar dapat memahami konsep atau teori penyelesaian permasalahan penanganan keluhan pelanggan

BAB 3 METODE PENELITIAN

Pada bab ini dijelaskan mengenai langkah-langkah penelitian beserta metodeyang digunakan. Langkah-langkah penelitian dijelaskan dalam sebuah diagram alur yang sistematis dan dijelaskan tahap demi tahap.

BAB 4HASILPENGUJIAN DAN ANALISIS DATA

Pada bab ini dilakukan pengujian terhadap metode penggalian teks berdasarkan skenario pengujian yang telah dirancang sebelumnya. Selain itu pada bab ini juga dijelaskan mengenai analisis hasil pengujian.

BAB 5KONTRIBUSI PENELITIAN

Bab ini berisi kontribusi penelitian yang terbagi menjadi kontribusi keilmuan dan kontribusi praktis.

BAB 6 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian ini dan juga saran bagi penelitian mendatang yang berasal dari temuan penelitian ini.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA DAN LANDASAN TEORI

Bab ini membahas tentang dasar teori yang berkaitan dengan penyusunan penelitian yang dilakukan serta membahas beberapa penelitian sebelumnya. Sedangkan dasar teori yang dibahas diantaranya konsep tentang data mining, *text mining*, *preprocessing* dalam text mining yang meliputi *case folding, filtering dan stemming,* metode *clustering* yang akan digunakan, serta metode evaluasi hasil *stemming* dan *clustering.* Selain itu, pada bab ini juga dibahas kontribusi dari hasil penelitian ini.

2.1 Kajian Pustaka

Pada bagian kajian pustaka ini, dibahas beberapa penelitian terdahulu yang berkaitan, yaitu penelitian-penelitian tentang keluhan pelanggan, dan penelitian-penelitian tentang penggunaan metode penggalian teks untuk kasus keluhan pelanggan.

2.1.1 Keluhan Pelanggan

Terdapat banyak penelitian yang membahas tentang keluhan pelanggan. Menurut Faed dkk(2015), keluhan pelanggan merupakan hal yang tidak bisa dihindari oleh perusahaan dan harus diperhatikan oleh perusahaan. Banyak peneliti telah menginvestigasi dan mengembangkan metode penanganan keluhan walaupun banyak *framework* dan metode pendekatan yang tidak berhasil dalam menangani fakta dimana perusahaan-perusahaan menghadapi jumlah keluhan yang tidak terhitung banyaknya. Hal tersebut disebabkan oleh ekspektasi yang tinggi terhadap perusahaan sehingga perusahaan tidak dapat menangani keluhan dengan efektif. Lebih lanjut, tidak seluruh keluhan ditangani oleh perusahaan. Dalam penelitian tersebut (Faed dkk, 2015), peneliti mengambil fokus pada keluhan-keluhan pelanggan yang harus ditangani dalam mempertahankan produktivitas dalam industri logistik dan transport, selain itu juga keluhan-keluhan yang perlu ditangani untuk dapat mempertahankan pelanggan dan mendapatkan pelanggan baru. Untuk menangani permasalahan tersebut, peneliti menggunakan *customer relatinship managemen framework,* sedangkan media pengumpulan data

yang digunakan adalah dengan kuisioner, sedangkan area pengumpulan data adalah Australia bagian barat. Hasil penelitian menunjukkan bahwa *framework* yang digunakan dapat mengatasi permasalahan sesuai ekspektasi pelanggan (Faed dkk, 2015).

Penelitian lain (Cambra-Fierro dkk, 2015) yang membahas keluhan pelanggan yang berhubungan dengan aktifitas-aktifitas penting yang menjadi fokus dalam penanganan keluhan pelanggan, yaitu *timeliness, compensation,* dan *communication.Timeliness* diartikan sebagai seberapa cepat perusahaan dalam merespon dan menyelesaikan keluhan pelanggan. Respon yang cepat akan memberikan keuntungan ekonomis, yaitu dengan meningkatkan efisiensi proses penanganan keluhan dan dapat menghemat sumber daya perusahaan. Selain itu, respon yang cepat juga memberikan keuntungan sosial karena pelanggan akan merasa bahwa perusahaan peduli dengan mereka dan dapat meningkatkan kepercayaan pelanggan terhadap perusahaan. Aktifitas yang kedua adalah *compensation,* yang meliputi *refund,* perbaikan produk, atau penggantian produk/*replacement* yang diberikan oleh perusahaan kepada pelanggan apabila terjadi kerusakan atau ditemukan cacat pada produk yang dijual. *Compensation* memberikan keuntungan ekonomis bagi pelanggan karena akan mengimbangi kerugian yang dialami oleh pelanggan yang disebabkan oleh produk yang cacat. *Communication* didefinisikan sebagai aktifitas permintaan maaf dan pemberian penjelasan oleh pihak perusahaan terhadap kesalahan yang terjadi kepada pelanggan disertai dengan solusi untuk menyelesaikan permasalahan tersebut. Untuk mengatasi hal tersebut peneliti mengembangkan framework dan menguji hipotesis tentang aktifitas-aktifitas penting tersebut. Data yang digunakan dalam penelitian tersebut adalah data yang didapatkan dari bank basar di benua Eropa. metode pengumpulan data yang digunakan adalah menggunakan kuisioner dan wawancara. Hasil penelitian menunjukkan bahwa framework yang dikembangkan sesuai dengan kondisi yang ada.

Aktifitas-aktifitas yang disebutkan pada penelitian Cambra-Fierro dkk (2015) berelasi dengan lima hal penting yang perlu dilakukan dalam melakukan *service recovery* (Bell and Zemke, 1987, dalam Mupemhi dkk, 2006). *Service recovery* sendiri didefinisikan sebagai proses untuk menuntun apabila perusahaan

gagal dalam memberikan pelayanan. Kelima hal tersebut adalah *apology, urgent reinstatement, emphaty, symbolic atonement,* dan *follow-up. Apology* adalah permintaan maaf dari penanggung jawab terhadap konsumen atas kegagalan layanan. *Urgent reinstatement* adalah kecepatan untuk mengambil tindakan, dalam hal ini kecepatan dalam merespon dan mengambil tindakan jika terjadi *service failure. Emphaty* adalah menunjukkan ekspresi dengan tulus dan mengerti perasaan konsumen yang mendapatkan layanan yang tidak memuaskan. *Symbolic attonement* adalah pemberian kompensasi, misalnya dengan memberikan layanan gratis, potongan harga, pengalihan ke layanan lain, maupun ganti rugi. *Follow-up* adalah tindakan yang dilakukan setelah melakukan *recovery* untuk memastikan pelanggan telah puas dengan proses *recovery* yang telah dilakukan.

Berdasarkan langkah-langkah untuk melakukan *service recovery,* Pei-wul dan Yan-qiu(2006) melakukan penelitian untuk mengetahui dampak dari dilakukannya *service recovery.* Secara spesifik penelitian tersebut dilakukan untuk mengetahui apakah ada pengaruh signifikan terhadap pelanggan yang memberikan komplain dan pelanggan yang tidak memberikan komplain setelah mereka diberikan *service recovery* yang sama. Dari hasil percobaan, dilakukan perbandingan respon dari kedua kelompok tersebut. Peneliti menggunakan metode pengujian hipotesis, dengan tujuh hipotesis yang akan diuji yaitu: H1, apakah pelanggan akan melakukan komplain atau tidak setelah mendapatkan layanan yang tidak sesuai, kepuasan mereka dan keinginan untuk kembali menggunakan layanan menurun. H2, kepuasan dan niat untuk kembali menggunakan layanan jauh menurun pada pelanggan yang komplain, dibandingkan pelanggan yang tidak komplain. H3, pelanggan yang komplain dan tidak komplain akan mengalami peningkatan kepuasan setelah diberikan *high level service recovery.* H4, pelanggan yang komplain maupun tidak, akan mengalami penurunan tingkat kepuasan dan keinginan menggunakan layanan lagi, setelah diberikan *low level service recovery.* H5, setelah diberikan *high level service recovery,* pelanggan yang melakukan komplain tidak banyak mengalami peningkatan kepuasan dan keinginan unutuk kembali menggunakan layanan, dibandingkan dengan pelanggan yang tidak komplain. H6, setelah diberikan *low level service recovery,* pelanggan yang melakukan komplain lebih banyak mengalami penurunan

kepuasan dan keinginan untuk kembali menggunakan layanan, dibandingkan dengan pelanggan yang tidak komplain. H7, pelanggan yang komplain lebih rendah "*recovery*"nya dibandingkan pelanggan yang tidak komplain. *High level service recovery* dalam penelitian tersebut dicontohkan sebagai permohonan maaf oleh manajer atas pelayanan yang kurang memuaskan dengan sikap yang sangat sopan, kemudian memberikan jaminan bahwa perusahaan (dalam kasus ini sebuah rumah makan) akan melakukan perbaikan dan memberikan makanan gratis. *Low level service recovery* dicontohkan sebagai permohonan maaf oleh pelayan, dengan sikap yang kurang baik, manajer melimpahkan kesalahan ke pihak lain dan mengatakan bahwa kesalahan yang terjadi adalah hal yang biasa. Metode pengumpulan data yang digunakan adalah metode kuisioner dengan 100 kuisioner untuk *high level* dan 100 kuisioner untuk low level. Dari jumlah tersebut didapatkan kuisioner valid sebanyak 177 buah. Hasilnya menunjukkan bahwa H1, H2, H3, dan H4 diterima. Kesimpulan yang didapatkan diantaranya bahwa pemberian *service recovery* yang sesuai sesaat setelah terjadi kegagalan layanan akan meningkatkan kepuasan pelanggan. Walaupun pelanggan yang melakukan komplain tampak lebih tidak puas daripada pelanggan yang tidak komplain, tetapi kepuasan mereka lebih cepat ditingkatkan kembali dengan *high level service recovery* dibandingkan pelanggan yang tidak komplain. *Low level service recovery* dapat menjadi lebih buruk dampaknya jika diberikan kepada pelanggan yang tidak komplain (Pei-wul dan Yan-qiu,2006).

2.1.2 Penggalian Teks untuk Keluhan Pelanggan

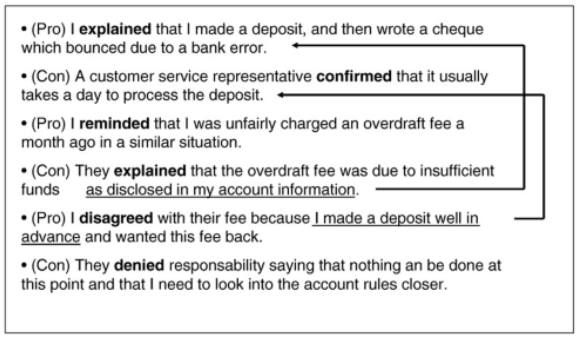
Pemanfaatan penggalian teks untuk menangani keluhan pelanggan telah dilakukan sebelumnya oleh Coussement dan den Poel (2008), Galitsky dkk, (2009). Trappey dkk, (2012), dan Dong dan Wang (2015). Coussement dan den Poel (2008) menggunakan metode penggalian teks untuk meningkatkan manajemen keluhan. Peneliti menggunakan metode klasifikasi teks untuk membedakan email keluhan dan email bukan keluhan. Alasan peneliti menggunakan email karena email menjadi salah satu media pelanggan dalam berinteraksi dengan perusahaan yang semakin banyak digunakan selain media komunikasi lain seperti telepon. Interaksi yang dilakukan pelanggan melalui email

diantaranya adalah pengaduan keluhan, sehingga informasi tentang keluhan pelanggan harus dapat digali dari email-email yang masuk. Penggalian informasi keluhan harus dapat dilakukan dengan efisien agar *service recovery* yang tepat dapat diberikan. Penggalian teks digunakan sebagai alat untuk menglasifikasi email keluhan secara otomatis. Pemanfaatan klasifikasi email secara otomatis memiliki keuntungan yaitu menghemat waktu, dan lebih murah biayanya dibandingkan dengan cara manual. Keuntungan yang lainnya adalah dengan mampu mengenali email keluhan dengan cepat, akan mengoptimasi proses penanganan keluhan. Dengan memisahkan email keluhan dan email bukan keluhan, perusahaan dapat memisahkan call center menjadi badan khusus yang menangani keluhan dan badan lain untuk menangani bukan keluhan, hal ini memungkinkan penanganan keluhan yang lebih fokus dan optimal. Pada penelitian tersebut, digunakan pendekatan baru yaitu *linguistic style* dalam klasifikasi email keluhan selain menggunakan pendekatan *vector space model* yang umum digunakan pada klasifikasi teks. *Liguistic style* digunakan sebagai fitur tambahan untuk melakukan klasifikasi dengan cara mengekstrak kata yang memiliki keunikan linguistik. Daftar kata linguistic telah didefinisikan sebelumnya, diantaranya *yes, ok , mmhmm* untuk persetujuan, *a, an, the* untuk artikel, dan *hour,day, o'clock* untuk indikasi waktu. Metode penggalian teks yang dilakukan dalam penelitian tersebut yaitu pra-proses yang meliputi *raw text cleaning, tokenization, case conversion, pos tager, stemming,* dan *term filtering.* Kemudian dilakukan *term weighting* dan *dimension reduction.* Langkah terakhir adalah melakukan klasifikasi teks keluhan dengan metode Adaboost. Untuk evaluasi, pada penelitian tersebut digunakan metode *confusion matrix*. Kesimpulan dari penelitian tersebut adalah, pemanfaatan metode penggalian teks untuk mengeksrak email keluhan memiliki tingkat keberhasilan 83%. Jika dibandingkan dengan mengekstrak email keluhan secara manual (diasumsikan biaya pegawai untuk 1 jam=29 Euro), dapat diprediksikan perusahaan dapat menghemat biaya 25.175 Euro dalam periode 5 tahun. Kesimpulan yang kedua adalah dengan memanfaatkan *linguistic style* sebagai salah satu fitur untuk klasifikasi, dapat disimpulkan bahwa email yang memiliki bentuk lampau (*past tense*) memiliki kemungkinan lebih besar sebagai email keluhan, dibandingkan

email dengan bentuk *future tense.* Selain itu, email yang memiliki kata negasi *no, never, not* juga memiliki kemungkinan yang lebih besar sebagai email keluhan oleh Coussement dan den Poel (2008).

Jika sebelumnya keluhan pelanggan diklasifikasikan dengan bantuan *linguistic style*, penelitian yang dilakukan oleh Galitsky dkk, (2009) menggunakan pendekatan yang berbeda. Jika pada umumnya klasifikasi teks menggunakan *vector space model*(VSM), maka pada penelitian tersebut tidak hanya berdasarkan VSM tetapi diusulkan pendekatan baru yaitu klasifikasi keluhan pelanggan menggunakan pendekatan "interaksi konsumen-perusahaan". Alasan diusulkannya metode tersebut karena banyaknya hambatan dalam menganalisis teks keluhan pelanggan, diantaranya struktur logika yang komplek, tidak konsisten, representasi yang tidak jelas (*biased*), dan tingginya ambiguitas. Untuk mengatasi hal tersebut, sebuah perangkat lunak bernama *ComplaintEngine* memiliki form khusus untuk mengadukan keluhan sehingga informasi yang relevan dalam keluhan dapat lebih mudah digali.

Pendekatan "interaksi konsumen-perusahaan" merupakan metode yang berdasar pada dialog yang terjadi antara pelanggan yang memberikan keluhan dan perusahaan yang memberikan respon dan disebut sebagai *complaint scenario. Complaint scenario* dimodelkan dalam bentuk *graph* yang menggambarkan aksi komunikasi mengenai situasi konflik yang terjadi, juga menggambarkan aliran interaksi yang terjadi diantara pelanggan dan pihak perusahaan. Aksi komunikasi pada penelitian tersebut didefinisikan sebagai kata yang mewakili karakter dari suatu interaksi antara pelanggan dan perusahaan dalam sebuah *complaint scenario.* Aksi-aksi komunikasi (*comunicative actions*) dikelompokkan menjadi enam, yaitu (*pro*)*explain, confirm, remind, (con)explain, disagree,* dan *deny.* Contoh *complain scenario* pada penelitian tersebut pada Gambar 2.1. *Complaint scenario* tersebut nantinya akan diklasifikasikan menjadi skenario yang valid dan skenario yang tidak valid.



Gambar 2.1 *Complain Scenario* (Galitsky dkk, 2009)

Data pada penelitian tersebut dikumpulkan dari website *PlanetFeedBack.com*. Untuk kepentingan penelitian tersebut, setiap komplain dilabeli sesuai urutan dalam *comunicative actions* secara manual oleh seorang ahli. Setelah data diberikan label, dilakukan pra-proses data, diantaranya membuang data yang tidak memenuhi kriteria. Contoh data yang tidak memenuhi kriteria adalah data yang didalamnya tidak ada interaksi dengan pihak perusahaan, misalnya pelanggan mengeluhkan suatu produk yang cacat, tetapi pihak perusahaan tidak memberi tanggapan. Untuk kasus tersebut, tidak bisa dianalisis dengan metode yang diusulkan. Metode klasifikasi yang digunakan adalah *supervised learning* (Galitsky dkk, 2009).

Kesimpulan dari penelitian tersebut adalah bahwa klasifikasi *complaint scenario* yang menggunakan *comunicative actions* memiliki hasil akurasi yang lebih tinggi 22% daripada yang tidak menggunakan *comunicative actions*, yaitu sekitar 50%. Pengembangan kedepan adalah pengintegrasian dengan *argument-based extension of logic programing*, dan peningkatan mekanisme pencocokan untuk klasifikasi *complaint scenario* (Galitsky dkk, 2009).

Penelitian ketiga yang dibahas dalam kajian literatur ini dilakukan oleh Trappey dkk (2012) tentang penggalian informasi mengenai kepuasan dan ketidak puasan pelanggan menggunakan metode analisis teks berbasis ontologi dengan *critical incident* (CI). Tujuan dari penelitian tersebut adalah menggunakan metode untuk memproses teks dialog pelanggan secara otomatis, membentuk *cluster* dialog, kemudian dari *cluster* tersebut dilakukan penarikan kesimpulan tentang *trend*, dan interpretasi kepuasan maupun ketidak puasan pelanggan. Studi kasus pada penelitian tersebut adalah tentang angkutan massa (*mass transit system*) di

Kaohsiung, Taiwan. Alasan peneliti melakukan penelitian tersebut adalah operator *mass transit system* saat mengevaluasi kepuasan pelanggan hanya berdasarkan ide secara umum, tetapi prasangka mereka dapat berbeda-beda, terbatas, bias, dan dapat sangat berbeda dengan faktor-faktor sebenarnya yang mempengaruhi kepuasan dan ketidak puasan pelanggan. Data yang digunakan untuk penelitian tersebut dikumpulkan dengan menyebarkan formulir yang diisi dengan pengalaman pelanggan dalam menggunakan jasa *mass transit system.* Pelanggan diarahkan untuk menceritakan pengalaman menyenangkan maupun pengalaman yang tidak menyenangkan pada formulis tersebut (Trappey dkk, 2012).

Metode yang digunakan peneliti pertama-tama adalah mendeskripsikan kata-kata unik yang berhubungan tentang *mass transit system.* Kemudian formulir dari pelanggan yang sudah dikumpulkan diproses menggunakan metode penggalian teks standar, diantaranya melakukan *phrasing* atau *tokenizing* untuk memecah *phrase* menjadi kata, *stop words* untuk menghapus kata yang tidak relevan, dan pembentukan *vector space model* dari kata. Proses selanjutnya adalah ekstraksi kata-kata kunci yang relevan dengan *mass transit system.* Langkah terakhir adalah melakukan pengelompokan data menggunakan metode *cluster k-means.* Hasil pengelompokan dari metode *k-means* adalah 5 kelompok keluhan yaitu kelompok pertama, pelanggan yang mengeluhkan kenyamanan dan kursi yang sempit; kelompok kedua mengeluhkan tingginya harga untuk membeli *metro transit card*; kelompok ketiga mengeluhkan suasana yang tidak nyaman karena keramaian pelanggan lain; kelompok keempat mengeluhkan repotnya berjalan di setasiun MRT (*mass rapid transportation*); dan kelompok kelima mengeluhkan tentang MRT. Peneliti menganggap bahwa kata yang terkumpul pada kelompok kelima masih terlalu umum, sehingga hanya dilabeli "MRT dan fitur-fiturnya"(Trappey dkk, 2012).

Kesimpulan dari penelitian tersebut diantaranya sebagian besar ketidak puasan pelanggan disebabkan karena terbatasnya kapasitas tempat duduk, dan perjalanan menjadi kurang nyaman terutama saat *peak season.* Hasil dari penelitian tersebut kedepannya diharapkan dapat menjadi dasar untuk membangun sistem penanganan keluhan pelanggan *mass transit system* yang lebih terstruktur (Trappey dkk, 2012).

Penelitian terkait terakhir yang dibahas disini adalah penelitian oleh Dong dan Wang (2015), yang membahas tentang evaluasi kualitas pelayanan dengan pengelompokan keluhan pelanggan. Peneliti melakukan klasifikasi keluhan pelanggan menjadi dua kelas, yaitu kelas yang pertama adalah servis yang berhasil memenuhi keluhan pelanggan dan meningkatkan kepuasan pelanggan, dan kelas kedua adalah servis yang tidak sesuai dengan yang diinginkan pelanggan. Tujuan peneliti melakukan penelitian tersebut adalah mencari pendekatan baru untuk mengevaluasi kualitas penanganan keluhan pelanggan oleh perusahaan. Data yang digunakan pada penelitian tersebut didapatkan dari catatan-catatan penanganan keluhan pelanggan dari perusahaan insurance di China pada tahun 2011 (Dong dan Wang, 2015).

Metode yang digunakan adalah metode penggalian teks dengan beberapa proses yaitu segmentasi dokumen, pemilihan fitur, pembentukan *vector space model*, dan terakhir proses klasifikasi. Metode klasfikasi yang digunakan pada penelitian tersebut adalah *support vector model* (SVM) dengan bantuan *library* LIBSVM. Sedangkan perangkat lunak yang digunakan dibangun dengan pemrograman *java*. Proses segmentasi dokumen dilakukan untuk memisahkan antara data yang akan digunakan untuk membangun model klasifikasi dan data yang digunakan untuk pengujian. Proses pemilihan fitur, terdiri dari tiga tahap yaitu mengekstraksi kata-kata dari dokumen, melakukan perhitungan *term frequency* (TF), melakukan perhitungan CHI statistik, menghapus kata yang memiliki nilai CHI rendah. Hasil dari pemilihan fitur tersebut dibentuk menjadi *vector space model*. Untuk data uji dilakukan proses yang sama, setelah didapatkan *vector space model*. Selanjutnya data diklasifikasikan dengan metode SVM. Kernel SVM yang digunakan adalah RBF kernel. Hasil pengujian untuk 923 dimensi vektor menunjukkan akurasi *cross validation* 66,89%, dan *test accuracy* 71,79%; untuk 399 dimensi vektor menunjukkan akurasi *cross validation* 75,5%, dan *test accuracy* 86,84%; untuk 42 dimensi vektor menunjukkan akurasi *cross validation* 84,1%, dan *test accuracy* 89,74%; 23 dimensi vektor menunjukkan akurasi *cross validation* 87,18%, dan *test accuracy* 94,87% (Dong dan Wang, 2015).

Kesimpulan dari penelitian tersebut bahwa penggunaan metode klasifikasi teks untuk mengklasifikasikan dokumen penanganan keluhan pelanggan terbukti efektif dalam mengevaluasi kualitas manajemen *customer service.* Penelitian ini dapat dikembangkan untuk menilai efektifitas keluhan pelanggan (Dong dan Wang, 2015).

2.2 Profil Perusahaan

Perusahaan Listrik Negara (PLN), merupakan perusahaan yang dimiliki pemerintah Republik Indonesia yang bergerak di bidang ketenagalistrikan di Indonesia. Perkembangan ketenagalistrikan di Indonesia berawal dari beberapa perusahaan pabrik gula dan pabrik teh milik pemerintah kolonial Belanda,yang mendirikan pembangkit listrik untuk keperluan mereka pada akhir abad ke 19. Pengelolaan pembangkit listrik tersebut sempat beralih kepada Jepang setelah Belanda menyerah kepada Jepang di awal Perang Dunia II. Setelah kekalahan Jepang pada tahun 1945, para pemuda dan buruh listrik melalui delegasi Buruh/Pegawai Listrik dan Gas yang bersama-sama dengan Pimpinan KNI Pusat berinisiatif menghadap Presiden Soekarno untuk menyerahkan perusahaan-perusahaan tersebut kepada Pemerintah Republik Indonesia. Kemudian Presiden Soekarno membentuk Jawatan Listrik dan Gas di bawah Departemen Pekerjaan Umum dan Tenaga dengan kapasitas pembangkit tenaga listrik sebesar 157,5 MW. Pada tanggal 1 Januari 1965, Jawatan Listrik dan Gas dipisahkan menjadi dua badan usaha milik negara, yaitu Perusahaan Listrik Negara (PLN) sebagai pengelola tenaga listrik milik negara dan Perusahaan Gas Negara (PGN) sebagai pengelola gas.

Saat ini PLN memiliki lima jenis pembangkit listrik, yaituyang menggunakan tenaga air (PLTA), tenaga uap (PLTU), tenaga gas (PLTG), tenaga panas bumi (PLTP), dan tenaga matahari/surya (PLTS) dengan kapasitas terpasang nasional sebesar 50.000MW yang dibangun PLN bersama pihak swasta sejak PLN berdiri. Sedangkan jumlah pegawai PLN saat ini kurang lebih berjumlah 49.000 orang. PLN memiliki kantor pusat di jalan Trunojoyo Blok M 1/135, Kebayoran Baru, Jakarta Selatan (www.pln.co.id).

2.2.1 Visi dan Misi

PLN memiliki visi dan misi perusahaan (www.pln.co.id) sebagai berikut :

Visi

Diakui sebagai Perusahaan Kelas Dunia yang Bertumbuh kembang, Unggul dan Terpercaya dengan bertumpu pada Potensi Insani.

Misi

- Menjalankan bisnis kelistrikan dan bidang lain yang terkait, berorientasi pada kepuasan pelanggan, anggota perusahaan dan pemegang saham.
- Menjadikan tenaga listrik sebagai media untuk meningkatkan kualitas kehidupan masyarakat.
- Mengupayakan agar tenaga listrik menjadi pendorong kegiatan ekonomi.
- Menjalankan kegiatan usaha yang berwawasan lingkungan.

2.2.2 Pelayanan Online

Situs resmi PLN memiliki beberapa layanan online untuk pelanggan yaitu riwayat prepaid pelanggan, cek tagihan rekening, penyambungan baru, perubahan daya, penyambungan sementara, listrik pintar, tarif tenaga listrik, uang jaminan langganan, QA seputar tarif listrik, dan informasi P2TL. Riwayat *prepaid* pelanggan merupakan layanan informasi untuk listrik prabayar, pada layanan ini pelanggan mendapatkan informasi riwayat pembelian token PLN dan transaksi token kwh non tunai. Cek tagihan rekening merupakan layanan pengecekan tagihan rekening pelanggan dengan memasukkan nomor ID pelanggan. Penyambungan baru merupakan layanan untuk pelanggan yang ingin mengajukan permintaan penyambungan listrik PLN. Pelanggan diminta memasukkan data mengenai lokasi yang ingin diberikan sambungan listrik, serta jumlah daya dan peruntukan penyambungan daya listrik tersebut. Perubahan daya merupakan layanan untuk memohon kenaikan atau penurunan daya listrik yang telah ada, sedangkan Sambung sementara adalah layanan untuk memohon penyambungan daya listrik untuk rentang waktu yang diinginkan pelanggan. Listrik pintar berisi informasi tertulis tentang penggunaan listrik prabayar. Tarif tenaga listrik berisi informasi tentang harga daya listrik per kwh (www.pln.co.id).

2.2.3 Keluhan Pelanggan

Keluhan pelanggan pada PLN dapat diajukan kepada call center PLN yang bernama PLN 123. Selain melalui call center, keluhan pelanggan dapat diajukan melalui website PLN, email, dan media sosial yaitu twitter dan facebook (www.pln.co.id).

Pada penelitian ini, data yang dikumpulkan dari media sosial twitter karena beberapa pertimbangan. Pertama, karena twitter pln (pln_123) secara resmi digunakan untuk menampung keluhan pelanggan PLN. Kedua, karena twitter bersifat terbuka sebagai media sosial, artinya siapa saja boleh mengambil informasi dari twitter. Ketiga, karena jumlah karakter yang dapat dituliskan terbatas yaitu 140 karakter, jumlah karakter yang terbatas akan memudahkan memproses data.

2.3 Data Twitter

Twitter merupakan salah satu media sosial yang banyak digunakan oleh pengguna internet di dunia. Pada tahun 2013 tercatat bahwa Indonesia menempati peringkat ketiga di dunia dalam hal penggunaan jejaring media twitter (Insaf 2013). Twitter memberikan kebebasan kepada penggunanya untuk mengirim pesan kepada *followers*-nya.

Karena kebebasan dalam mengirim pesan, maka struktur bahasa yang digunakan tidak selalu mengikuti ejaan bahasa yang benar. Hal tersebut menyebabkan struktur bahasa yang ada pada twitter menjadi semi-terstruktur. Menurut Naraadhipa dan Purwarianti (2011) ada empat bentuk penyimpangan ejaan bahasa Indonesia yang umum terdapat di twitter yaitu penulisan angka diawal atau diakhir kata, penggantian huruf dengan angka, perulangan karakter, dan kata tidak baku. Penyimpangan ejaan tersebut menjadi *noise* saat melakukan pemrosesan teks.

Selain penyimpangan ejaan, data teks pada twitter memiliki ciri khusus yaitu simbol @ dan #. Simbol @ merupakan simbol untuk nama pengguna dan biasanya digunakan untuk *retweet* atau mengirim pesan yang sama kepada pengguna lain. Simbol # dikenal sebagai “hash-tag”, tag yang dimaksud adalah topik dimana pesan tersebut ditujukan. Ciri khusus pada twitter ini sering kali

diabaikan, tetapi simbol-simbol tersebut sebenarnya dapat memiliki makna tertentu, sehingga perlu dilakukan penanganan (Oussalah dkk, 2016).

2.4 Data Mining dan Fungsinya

Data mining didefinisikan sebagai proses yang bertujuan untuk menemukan hubungan atau pola dari tempan penyimpanan data dengan menggunakan teknik pengenalan pola (Kusnawi, 2007). Pendapat lain mengatakan bahwa definisi data mining adalah proses menemukan pengetahuan yang menarik dari data-data yang tersimpan dalam *database.*(Han dan Kamber, 2006)

Fungsi data mining (Han dan Kamber, 2006) adalah sebagai berikut:

1) Menggambarkan pola atau kecenderungan yang sering muncul dari sebuah data.
2) Melakukan estimasi yaitu mengklasifikasikan data numerik menurut kelasnya.
3) Melakukan prediksi atau *forecasting* untuk memperkirakan nilai pada waktu yang akan datang.
4) Melakukan klasifikasi.
5) Melakukan *clustering* yang mengacu pada pengelompokan *record*, observasi, atau kasus kedalam objek yang sama.
6) Menemukan asosiasi atau menemukan pola pada suatu data.

Adapun proses menemukan pengetahuan meliputi data *selection, preprocessing, transformation, data mining* dan *evaluation*(Kusnawi, 2007).

1) Data Selection

Data Selection atau seleksi data dilakukan sebelum tahap mengolah informasi, kemudian data disimpan pada suatu berkas data.

2) *Preprocessing*

Proses *Preprocessing* data merupakan proses untuk meningkatkan kualitas data yang digunakan, karena pada proses ini dilakukan penghapusan *noise.* Kemudian proses *cleaning* yaitu membuang duplikasi data, dan memeriksa data yang tidak konsisten, sehingga akan menghasilkan akurasi lebih baik dan mengurangi *cost* sistem.

3) *Transformation*

Proses *transformation* dilakukan dengan mencari fitur-fitur yang berguna dalam mempresentasikan data sehingga sesuai untuk proses *data mining*. Ini merupakan proses kreatif dan bergantung pada jenis atau pola informasi yang akan dicari.

4) *Data Mining*

Proses *data mining* yaitu proses mencari pola atau informasi dalam data terpilih dengan menggunakan teknik atau metode tertentu. Pemilihan metode yang tepat bergantung pada tujuan dan proses pencarian informasi.

5) *Evaluation*

Tahap ini merupakan bagian dari proses pencarian informasi yang mencakup pemeriksaan kesesuaian pola atau informasi yang dihasilkan dengan fakta atau hipotesa yang ada.

2.5 Text Mining

Text mining adalah metode untuk mencari pola pada teks yaitu dengan melakukan analisis untuk mengambil informasi yang berguna untuk tujuan tertentu. *Text mining* adalah salah satu bidang khusus dari *data mining*. Tujuan dari text mining adalah untuk mendapatkan informasi yang dibutuhkan dari dokumen. Sumber data yang digunakan adalah kumpulan teks yang tidak terstruktur atau semi terstruktur, berbeda dengan *data mining* yang umumnya menggunakan data terstruktur. Tugas khusus dari *text mining* diantaranya pengkategorisasian teks dan pengelompokan teks (Feldman dan Sanger, 2007).

Text mining mampu memberikan solusi dari permasalahan pemrosesan, pengorganisasian, dan untuk menganalisa teks tidak terstruktur. *Text mining* menggunakan dan mengembangkan banyak teknik diantaranya *data mining,machine learning,* dan *information retrival.*

Karena *text mining* adalah salah satu bidang *data mining*, permasalahan yang dihadapi pun sama, yaitu jumlah data besar, data noise, dan perubahan struktur yang terus terjadi.

Proses yang dilakuan dalam *text mining* secara umum ada tiga yaitu *pre-processing, text transformation,* dan *pattern discovery.* Berikut ini akan dibahas

pre-procesing saja, karena *text transformation* akan dibahas pada sub bahasan tentang Pembobotan (sub-bab 2.9),dan *pattern discovery* akan dibahas pada sub bahasan tentang metode *clustering* (sub-bab 2.10).

Pre-processing

Pada tahap ini dilakukan pengolahan data untuk digunakan pada proses selanjutnya. Tujuan dilakukannya proses ini adalah untuk meningkatkan kecepatan sistem secara keseluruhan karena pada proses ini dilakukan penghapusan data yang tidak dibutuhkan. Tahap *pre-processing* ini terdiri dari *case folding, tokenizing, filtering,* dan *stemming.*

1) Tokenizing

Pada proses *tokenizing* dilakukan pemotongan-pemotongan kalimat menjadi kumpulan karakter yang lebih kecil yang disebut token.

2) Case Folding

Proses *case folding* adalah proses pengubahan karakter kapital menjadi non kapital. Selain itu karakter yang tidak dibutuhkan akan dibuang, karakter yang tidak dibutuhkan tersebut dikenal dengan *delimiter*.

3) Filtering

Proses *filtering* adalah proses memilih token mana yang harus digunakan untuk menggambarkan isi dokumen tersebut, sehingga jelas antara dokumen yang satu dengan yang lainnya. Proses yang dilakukan adalah mencari *stop words* pada token kemudian token tersebut akan dihapus. *Stop words* adalah daftar kata-kata yang tidak menggambarkan isi dokumen tersebut, misalnya kata penghubung.

4) Stemming

Proses *stemming* adalah proses untuk mengubah bentuk kata menjadi kata dasar. Cara kerjanya adalah dengan membuang imbuhan, sisipan, dan akhiran. Tujuan proses *stemming* diantaranya adalah meningkatkan efisiensi sistem. Proses *stemming* berkaitan dengan bahasa yang digunakan. Oleh sebab itu, algoritma *stemming* yang diterapkan untuk masing-masing bahasa akan berbeda.

Metode*Stemmer*

Algoritma *stemmer* terbagi menjadi dua jenis yaitu *stemmer* berbasis statistik (*statistic-based stemmer*) dan berbasis aturan (*rule-based stemmer*). *Statistic-based stemmer* merupakan metode *supervised* yang menggunakan data latih untuk membentuk model berupa parameter dan aturan dalam melakukan *stemming*, sedangkan *rule-based stemmer* merupakan metode *unsupervised* yang menggunakan sejumlah aturan yang sudah didefinisikan untuk melakukan *stemming*. Keunggulan dari *statistic-based stemmer* adalah dapat diterapkan pada bahasa yang berbeda dengan lebih mudah, memungkinkan untuk melakukan *stemming* multi bahasa. Kekurangan dari *statistic-based stemmer* adalah membutuhkan data dengan jumlah yang cukup untuk mendapat akurasi yang bagus sehingga kurang cocok untuk jumlah data sedikit, tidak dapat menangani kata yang diluar model statistik, contoh "sing-song", "foot-feet" (Brychcíndan Konopík, 2015). Keunggulan dari *rule-based stemmer* adalah memiliki akurasi yang tinggi, tidak membutuhkan data latih (*unsupervised*). Kekurangan dari *rule-based stemmer* adalah hanya dapat digunakan untuk satu bahasa, karena melakukan *stemming* untuk bahasa lain membutuhkan definisi aturan yang berbeda, oleh sebab itu sangat sulit untuk melakukan *stemming* multi bahasa.

1) Porter *Stemmer*

Porter Stemmer merupakan metode *rule-based stemmer* yang diciptakan oleh Porter pada 1980. Metode ini digunakan untuk mencari kata dasar Bahasa Inggris dengan cara menjalankan sejumlah aturan penghapusan dan penggantian huruf pada kata. Pada metode *Porter,* semua karakter selain A, I, U, E, O, dan Y dianggap sebagai konsonan, dengan pengecualian seperti pada kata TOY, maka konsonan adalah T dan Y. *S jika stem memiliki akhiran S, *v* jika stem mengandung huruf *vowel* *d jika stem diakhiri dobel konsonan, *o jika stem diakhiri pola konsonan-vokal-konsonan (cvc) selain huruf X, W, dan Y. Aturan yang diberlakukan pada metode *Porter*terdiri dari 5 langkah yaitu :

1. Ubah bentuk past participle dan plural

a. sses → ss, ies → i, s → empty

b. eed → ee, ed → empty, ing → empty

at → ate, bl → ble, iz → ize

2. Ubah bentuk :

ational →ate, tional → tion, enci → ence, anci → ance, izer → ize, abli → able, alli→ al, entli → ent, eli → e, ousli → ous, ization → ize, ation → ate , ator → ate, alism → al, iveness → ive, fulness → ful, ousness → ous, aliti → al, iviti → ive, biliti → ble.

3. Ubah bentuk :

icate → ic, active → *empty,* alize → al, iciti → ic, ical → ic, ful → empty, ness → empty

4. Ubah bentuk :

al → empty, ance → empty, ence → empty, er → empty, ic → empty, able → empty, ible → empty, ant → empty, ement → empty, ment → empty, ent → empty, ion → empty, ou → empty, ism → empty, ate → empty, iti → empty, ous → empty, ive → empty, ize → empty,

2) Nazief *Stemmer*

Algoritma stemming Nazief dan Adriani dikembangkan berdasarkan aturan bahasa Indonesia yang kata-katanya menggunakan imbuhan, yakni awalan(*prefix*), sisipan (*infix*), akhiran (*suffix*) dan kombinasi awalan dan akhiran(*confixes*). Nazief *stemmer* mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut:

a.*Inflection Suffixes* : kelompok akhiran yang tidak mengubah bentuk kata dasar. Kelompok ini dibagi menjadi dua, yakni: (1) *Particle* (Partikel) yangdidalamnya -*lah, -kah, -tah, dan -pun*; (2) *Passive Pronoun* (Kata ganti kepemilikan) yang didalamnya *-ku, -mu, -nya*.

b.*Derivation Suffixes* (Akhiran): kumpulan akhiran yang secara langsung ditambahkan pada kata dasar.

c.*Derivation Prefixes* (Awalan): Kumpulan awalan yang dapat ditambahkan langsung pada kata dasar yang sudah mendapatkan penambahan sampai dua awalan.

Langkah *Nazief stemmer* adalah sebagai berikut:

1. Untuk setiap kata, lakukan pengecekan terhadap kamus kata dasar. Jika ditemukan, kata dasar tersebut menjadi hasil keluaran proses dan proses berhenti, jika tidak ditemukan, proses dilanjutkan ke tahap berikutnya.
2. Hapus *Inflectional suffix* yaitu particle suffixes {lah, kah, tah, pun}, dan *possesive pronoun suffixes* {ku, mu, nya}. Lakukan pengecekan kamus kata dasar.
3. Hapus *Derrivational suffixes* {i, kan, an}. Lakukan pengecekan kamus kata dasar.
4. Hapus *Derrivational preffixes*{be, di, ke, me, pe, se, te}.
   - Hentikan proses jika :
     1. *preffixes* yang teridentifikasi membentuk kombinasi yang dilarang.
     2. *preffixes* yang teridentifikasi sama dengan *preffixes* yang sudah dihapus sebelumnya.
     3. tiga *preffixes* telah dihapus.
   - Identifikasi tipe *preffixes*
     1. *preffixes* {di, ke, se} dapat dihapus langsung.
     2. *preffixes* {be, te, me, pe} harus dilakukan pengecekan aturan.

Tabel 2.1Aturan Penghapusan Prefik dan Recoding

Nomor	Aturan	Hasil
1	berV	ber-V. . . | be-rV. . .
2	berCAP	ber-CAP. . . dimana C!='r' and P!='er'
3	berCAerV	ber-CAerV. . . dimana C!='r'
4	belajar	bel-ajar. . .
5	beC1erC2	be-C1erC2. . . dimana C!={'r'| 'l'}
6	terV	ter-V. . . | te-rV. . .
7	terCerV	ter-CerV. . . where C!='r'
8	terCP	ter-CP. . . where C!='r' and P!='er'
9	teC1erC2	te-CerC. . . where C!='r'
10	me{l|r|w|y}V	me-{l|r|w|y}V. . .
11	mem{b|f|v}	mem-{b|f|v}. . .
12	mempe{r|l}	mem-pe. . .
13	mem{rV|V}	me-m{rV|V}. . . | me-p{rV|V}. .
14	men{c|d|j|z}	men-{c|d|j|z}. . .
15	menV	me-nV. . . | me-tV. . .
16	meng{g|h|q}	meng-{g|h|q}. . .

Nomor	Aturan	Hasil
17	mengV	meng-V. . . | meng-kV. . .
18	menyV	meny-sV. . .
19	mempV	mem-pV. . . where V!='e'
20	pe{w|y}V	pe-{w|y}V. . .
21	perV	per-V. . . | pe-rV. . .
22	perCAP	per-CAP. . . where C!='r' and P!='er'
23	perCAerV	per-CAerV. . . where C!='r'
24	pem{b|f|v}	pem-{b|f|v}. . .
25	pem{rV|V}	pe-m{rV|V}. . . | pe-p{rV|V}. . .
26	pen{c|d|j|z}	pen-{c|d|j|z}. . .
27	penV	pe-nV. . . | pe-tV. . .
28	peng{g|h|q}	peng-{g|h|q}. . .
29	pengV	peng-V. . . | peng-kV. . .
30	penyV	peny-sV. . .
31	pelV	pe-lV. . .
32	peCerV	per-erV. . . where C!={r|w|y|l|m|n}
33	peCP	pe-CP. . . where C!={r|w|y|l|m|n} and P!='er'

3) Arifin Stemmmer

Algoritma ini (Arifin dan Setiono, 2002) didahului dengan pembacaan tiap kata dari *file* sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan:

1) Pemeriksaan semua kemungkinan bentuk kata. Setiap kata diasumsikan memiliki 2 awalan (*prefiks*) dan 3 akhiran (*sufiks*). Sehingga bentuknya menjadi: *Prefiks 1 +Prefiks 2 +Kata dasar +Sufiks 3 +Sufiks 2 +Sufiks 1*. Seandainya kata tersebut tidak memiliki imbuhan sebanyak imbuhan di atas, maka imbuhan yang kosong diberi tanda *x* untuk prefiks dan diberi tanda *xx* untuk sufiks.

2) Pemotongan dilakukan secara berurutan

sebagai berikut :

AW : AW

AK : AK

KD : KD

Dimana : AW= Awalan, AK= Akhiran, dan KD= Kata dasar

a. AW I, hasilnya disimpan pada p1

b. AW II, hasilnya disimpan pada p2

c. AK I, hasilnya disimpan pada s1

d. AK II, hasilnya disimpan pada s2

e. AK III, hasilnya disimpan pada s3

Pada setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Kalau pemeriksaan ini berhasil, maka proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya

3) Namun jika sampai pada pemotongan AK III, belum juga ditemukan di kamus, maka dilakukan proses kombinasi. KD yang dihasilkan dikombinasikan dengan imbuhan-imbuhannya dalam 12 konfigurasi berikut :

a. KD
b. KD + AK III
c. KD + AK III + AK II
d. KD + AK III + AK II + AK I
e. AW I + AW II + KD
f. AW I + AW II + KD + AK III
g. AW I + AW II + KD + AK III + AK II
h. AW I + AW II + KD + AK III + AKII + AKI
i. AW II + KD
j. AW II + KD + AK III
k. AW II + KD + AK III + AK II
l. AW II + KD + AK III + AK II + AK I

4) Fadillah *Stemmer*

Dalam metode *fadillah*, kata yang memiliki suku kata lebih dari dua dianggap belum kata dasar, sehingga perlu diproses dengan algoritma *stemmer*, tetapi apabila suku katanya telah berjumlah dua, maka dianggap kata dasar. Kelebihan dari metode *stemmer fadillah* ini adalah tidak memiliki ketergantungan dengan kamus kata dasar, tetapi kekuranganya metode ini terbatas pada jumlah aturan yang didefinisikan.

Metode *stemmer fadillah* menggunakan beberapa aturan *removal* atau aturan penggantian sebagai berikut (Tala, 2003):

Tabel 2.2 Kelompok Inflectional Particles

Suffix	Replacement	Measure Condition	Additional Condition	Examples
Kah	Null	2	Null	Tumbuhkah – tumbuh
Lah	Null	2	Null	Tumbuhlah – tumbuh
Pun	Null	2	Null	tumbuhpun - tumbuh

Tabel 2.2 menunjukkan imbuhan-imbuhan dalam kelompok *inflectional particles.* Dalam bahasa Indonesia, *inflectional particles* termasuk dalam akhiran, yaitu imbuhan yang letaknya berada di akhir kata.

Pada tabel 2.2, 2.3, 2.4, 2.5, dan 2.6 terdapat lima kolom, kolom pertama adalah *suffix* yaitu imbuhan. Kolom yang kedua adalah *replacement,* yang menunjukkan karakter pengganti untuk imbuhan saat dilakukan proses *stemming.* Kolom ketiga adalah *measure condition* yang menunjukkan jumlah suku kata minimal dari sebuah kata. Kolom keempat adalah *additional condition* menunjukkan kondisi khusus aturan *stemming* tersebut berlaku. Misalnya pada tabel 2.6 baris kedua, *additional condition :* Prefix !€ {di, meng, ter} , artinya aturan tersebut berlaku apabila awalan dari kata bukan salah satu dari {di, meng, ter}.

Tabel 2.3 Kelompok Inflectional Possessive Pronoun

Suffix	Replacement	Measure Condition	Additional Condition	Examples
Ku	Null	2	Null	Tubuhku – tubuh
Mu	Null	2	Null	Tubuhmu – tubuh
Nya	Null	2	Null	Tubuhnya - tubuh

Tabel 2.3 menunjukkan imbuhan-imbuhan dalam kelompok *inflectional possesive pronoun.* Dalam bahasa Indonesia, *inflectional possesive pronoun* termasuk dalam akhiran.

Tabel 2.4 Kelompok First Order of Derivational Prefixes

Suffix	Replacement	Measure Condition	Additional Condition	Examples
Meng	Null	2	Null	mengukur – ukur
Meny	s	2	V...	menyapu – sapu
Men	Null	2	Null	menduga – duga
Mem	p	2	V...	Memilah – pilah
Mem	Null	2	Null	Membaca – baca
Me	Null	2	Null	Merusak – rusak
Peng	Null	2	Null	Pengukur – ukur
Peny	s	2	V...	Penyapu – sapu
Pen	Null	2	Null	Penduga - duga
Pem	p	2	V...	Pemilah – pilah
Pem	Null	2	Null	Pembaca – baca
Di	Null	2	Null	Dilempar – lempar

Ter	Null	2	Null	Terlempar – lempar
Ke	Null	2	null	Kekasih - kasih

Tabel 2.4 menunjukkan imbuhan-imbuhan dalam kelompok *first order of derivational prefixes.* Dalam bahasa Indonesia, *first order of derivational prefixes* termasuk dalam awalan, yaitu jenis imbuhan yang letaknya di awal kata.

Tabel 2.5 Kelompok Second Order of Derivational Prefixes

Suffix	Replacement	Measure Condition	Additional Condition	Examples
Ber	Null	2	Null	Berjalan-jalan
Bel	Null	2	ajar	belajar – ajar
Be	Null	2	K er..	bekerja - kerja
Per	Null	2	Null	Perluas - luas
Pel	Null	2	Ajar	Pelajar – ajar
Pe	Null	2	Null	Pekerja - kerja

Tabel 2.5 menunjukkan imbuhan-imbuhan dalam kelompok *second order of derivational prefixes.* Dalam bahasa Indonesia, s*econd order of derivational prefixes* termasuk dalam awalan.

Tabel 2.6 Kelompok Derivational Suffixes

Suffix	Replacement	Measure Condition	Additional Condition	Examples
Ka.n	Null	2	Prefix !Є {ke, peng}	tarikkan – tarik (meng)ambilkan-ambil
an	Null	2	Prefix !Є {di, meng, ter}	Makanan - makan (per)janjian - janji
i	Null	2	V|K...c_1c_1, $c_1 \neq s$, $c_2 \neq i$ and prefix !Є {ber, ke, peng }	tandai – tanda (mandapati) - dapat

Tabel 2.6 menunjukkan imbuhan-imbuhan dalam kelompok *derivational suffixes.* Dalam bahasa Indonesia, *derivational suffixes* termasuk dalam akhiran.

Suku kata

Pada *rule-based stemmer* (Tala, 2003) digunakan suku kata untuk melakukan pengecekan kata dasar. Suku kata merupakan suatu struktur yang terjadi dari satu atau urutan fonem yang merupakan bagian kata dan ditandai oleh satu vokal, termasuk diftong. Diftong adalah dua vokal berurutan yang membentuk kasatuan, misalnya *au, ia, ue*. Sedangkan fonem ada yang terdiri dari dua konsonan yang berurutan, misalnya *ng, ny, sy*. Suku kata pada *rule-based stemmer* digunakan untuk melakukan pengecekan kata dasar. Suatu kata dianggap kata dasar apabila paling sedikit memiliki dua suku kata (Tala, 2003).

Pemenggalan suku kata digunakan dalam penelitian oleh Prasetyo dan Timoteus(2007) untuk menyiapkan data masukan dokumen LATEX.Tujuan

dilakukannya penelitian tersebut karena pemenggalan kata bahasa Indonesia masih belum sempurna, hal tersebut disebabkan karena penggunaan *data base* pemenggalan kata yang ada saat itu mengguakan dasar bahasa Melayu yang memiliki sedikit perbedaan dengan bahasa Indonesia. Peneliti tersebut melakukan modifikasi dan penyesuaian pemenggalan kata agar cocok dengan aturan bahasa Indonesia.

Peneliti lain yang menggunakan pemenggalan kata adalah Wasista dan Astin (2010), yang menggunakan metode pemenggalan kata bahasa Indonesia untuk mengubah teks menjadi suara. Masukan sistem adalah teks yang dibaca menggunakan kamera *web*, kemudian kata akan dikenali dan dilabeli urutan huruf vokal dan konsonannya. Setelah setiap huruf dilabeli dengan V untuk vokal, K untuk konsonan, peneliti memberikan label *n, y, g* untuk huruf *n,y,* dan *g*, dengan tujuan untuk memudahkan pemenggalan kata apabila ditemukan huruf konsunan yang berurutan. Hasil proses pelabelan tersebut menjadi masukan untuk proses pemenggalan suku kata. Pola umum suku kata dalam bahasa Indonesia sebagai berikut :

a) V a-nak, ba-u
b) VK an-da, da-un
c) KV se-bab, man-di
d) KVK lan-tai, ma-kan
e) KKV pra-ha-rai, sas-tra
f) KKVK frik-si, kon-trak
g) VKK eks, ons
h) KVKK pers, kon-teks
i) KKVKK kom-pleks
j) KKKV in-stru-men
k) KKKVK struk-tur

Untuk memenggal suku kata, digunakan pedoman berikut ini :

a. Apabila di tengah kata terdapat dua vokal berturutan (selain diftong), pemisahan dilakukan di antar kedua vokal tersebut.

b. Apabila di tengah kata terdapat konsonan di antara dua vokal, pemisahan dilakukan sebelum konsonan tersebut

c. Apabila di tengah kata terdapat dua konsonan atau lebih, pemisahan dilakukan setelah konsonan pertama

d. Imbuhan dan partikel yang biasanya ditulis serangkai dengan kata dasar, pada penyukuan dipisahkan

Setelah pemenggalan suku kata dilakukan, peneliti menyiapkan rekaman suara untuk setiap suku kata. Keluaran sistem tersebut berupa suara hasil pembacaan suku kata dari teks (Sigit dan Novita, 2010).

5) Asian *Stemmer*

Jelita Asian (2007) mengembangkan algoritma *naziefstemmer*, dengan menambahkan beberapa perbaikan yang bertujuan untuk meningkatkan hasil *stemming* yang diperoleh (Asian, 2007). Algoritma ini dikenal sebagai *confixstripping stemmer*.Perbaikan tersebut antara lain sebagaiberikut:

a.Menggunakan kamus kata dasar yang lebih lengkap

b.Memodifikasi dan menambahkan aturan pemenggalan untuk tipe awalan yang kompleks.

c. Menambahkan aturan *stemming* untuk kata ulang dan bentuk jamak, misalnya kata (1) 'buku-buku' yang menjadi 'buku', (2) „berkirim-kiriman‟menjadi kirim.

d. Mengubah urutan *stemming* untuk beberapa kasus tertentu. Algoritma *stemmer* yang diperkenalkan oleh Nazief akan menghilangkan akhiran lebih dahulu, baru diikuti penghapusan awalan. Namun menurut Asian, cara tersebut tidak selalu berhasil pada beberapa kata. Oleh sebab itu, diberikan aturan yang akan mengubah urutan *stemming*, yang mana penhapusan awalan dilakukan terlebih dahulu kemudian diikuti oleh penghapusan akhiran. Aturan ini disebut *rule precedence* dan berlaku jika kata memiliki kombinasi awalan-akhiran *be-lah, be-an, me-i, di-i,'pe-i*, atau *te-i.*

6) Enhanched Confix Stripping *Stemmer*

Metode *enhanced confis stripping stemmer* merupakan metode pengembangan *asian stemmer* yang dilakukan oleh (Arifin dkk, 2009). Peneliti tersebut menambahkan beberapa aturan yang belum ada pada Tabel 2.1 yaitu *mem+p*, untuk kata *mempromosikan, memproteksi, memprediksi*. *Men+s*, untuk

kata *mensyaratkan*, mensyukuri. *Menge+…*, untuk kata *mengerem*. *Penge+…*, untuk kata pengeboman. *Peng+k*, untuk kata *pengkajian*, dan kesalahan penghapusan *suffix*. Hasil penelitian menunjukkan metode tersebut berhasil menangani bentuk-bentuk imbuhan tersebut.

Selain penambahan aturan untuk penghapusan awalan, juga ditambahkan suatu tahap yang disebut *loop PengembalianAkhiran.* Tahap ini dilakukan setelah proses *recoding* gagal menemukan kata dasar. Pertama, metode akan mengembalikan akhiran/ sufik secara berurutan yaitu *derivational suffix*, kemudian *possesive pronoun*, dan terakhir adalah *inflrctional particle*. Setiap akhiran yang dikembalikan akan dicoba untuk dilakukan penghapusan awalan/prefik. Tahap terakhir adalah kembalikan seluruh akhiran dan lakukan tahap *recoding* sekali lagi, jika kata dasar masih belum ditemukan maka masukan awal dianggap sebagai kata dasar, dan seluruh proses dihentikan.

Tahap *loop PengembalianAkhiran* merupakan tahap yang dikembangkan dari kombinasi awalan-akhiran yang ada pada metode *arifin stemmer* karena sama-sama dikembangkan oleh peneliti yang sama. Selain itu, fungsi *loop PengembalianAkhiran* hampir sama dengan tahap kombinasi awalan-akhiran, yaitu untuk menghindari *overstemming.*

7) Arifiyanti *Stemmer*

*Stemmer*Arifiyanti merupakan pengembangan dari *asian stemmer* dengan memodifikasi imbuhan akhiran/sufikpada kelompok *derivation suffix* yakni *-in*, kemudian pada *inflectional suffix* yaitu imbuhan kepemilikan*–ny*. Selain itu dilakukan penambahan untuk partikel, yakni *–kh* dan *–pn.* Kedua partikel tersebut merupakan imbuhan *-kah* dan *-pun* yang paling sering dihilangkan huruf vokalnya. Imbuhan awalan juga ditambahkan aturan *d-* yang merupakan modifikasi dari *di-*. Selain itu juga terdapat tambahan kombinasi awal-akhiran pada *rule precedence*. Tambahan pada aturan *rule precedence* tersebut yaitu *di-in, se-i, n-i, n-a*, dan *te-k.*

Pada metode *arifiyanti stemmer* tidak ada perubahan dalam urutan tahap penghapusan imbuhan maupun penambahan tahap khusus seperti kombinasi

awalan-akhiran dengan kata dasar pada metode *arifin stemmer*. Seluruh tahap dan urutan sama dengan metode *asian stemmer.*

2.6 Penelitian Terkait tentang *Stemmer* dan Dampaknya pada Penggalian Informasi

Penelitian yang membahas tentang metode *stemmer* adalah yang dilakukan oleh (Floresdan Moreira, 2016). Pada penelitian tersebut dilakukan perbandingan metode stemming untuk empat bahasa dan dilakukan perbandingan akurasi metode *stemmer* dan pengaruh tiap metode terhadap hasil akhir *Information retrieval.* Metode *stemmer* yang digunakan adalah metode *stemmer* untuk bahasa Inggris, Perancis, Spanyol, dan Portugis. Sedangkan nama metode *stemmer* yang digunakan adalah Lovins, Paice/Husk, Porter, UEA-Lite, Linguistica, *Stemmer*-S, GRAS, dan Trunc3 sampai Trunc8. Untuk melakukan evaluasi metode *stemmer* digunakan empat variabel yaitu *Overstemming Index, Understemming Index, Stemming Weight*, dan *Error Rate Relative to Trunction*. Data yang digunakan dalam penelitian tersebut adalah artikel-artikel berita yang dikumpulkan dari Times, LeMondeandSDA, FolhadeSãoPaulo, dan AgenciaEFE. Untuk mengetahui pengaruh metode *stemmer* terhadap hasil *Information retrieval* dilakukan uji statistik *t* dengan α=0,05. Hasil pengujian menunjukan bahwa untuk bahasa Portugis, *stemmer* yang signifikan adalah UniNE, Porter, *Stemmer*-S, Trunc5, dan Trunc7. Untuk bahasa Perancis, *stemmer* yang signifikan adalah UniNE, Porter, RSLP Paice/Husk, GRAS, Trunc5, Trunc6, Trunc7, dan Trunc8. Untuk bahasa Spanyol, *stemmer* yang signifikan adalah UniNE, Porter, GRAS, Trunc5, Trunc6, Trunc7, dan Trunc8. Untuk bahasa Inggris, *stemmer* yang signifikan adalah *Stemmer*-S dan GRAS. Kesimpulan dari penelitian tersebut diantaranya adalah algoritma *stemmer* yang paling akurat ternyata bukan satu-satunya jalan untuk meningkatkan performa hasil *Information retrieval* pada semua bahasa, hasil eksperimen juga menunjukkan bahwa *light stemmer*yang lebih sederhana juga dapat memberikan performa yang bagus terhadap *Information retrieval*. Sebuah *stemmer* tidak harus mampu melakukan *stem* dengan tepat, tetapi jika *stemmer* mampu menangkap makna dari kata dengan baik, akurasi hasil penggalian informasi akan tetap tinggi (Flores dan Moreira, 2016).

Penelitian yang membahas tentang metode-metode *stemmer* untuk bebrapa bahasa di negara-negara Eropa juga dilakukan oleh (Brychcíndan Miloslav, 2015). Peneliti tersebut mengusulkan metode *stemmer* yang disebut High Precision *Stemmer* (HPS). HPS merupakan metode *supervised* yang terdiri dari dua tahap. Tahap pertama adalah mengelompokkan kata berdasarkan konteksnya dan setiap kelompok kata tersebut kemudian didefinisikan kata dasarnya. Tahap kedua adalah mengklasifikasikan kata yang akan dicari kata dasarnya kedalam kelompok-kelompok kata yang telah terbentuk. Tahap pertama berfungsi sebagai tahap *learning*, sehingga data latih dibutuhkan. Untuk metode *clustering* yang digunakan pada tahap pertama adalam *modified* MMI *clustering*, sedangkan metode klasifikasi yang digunakan pada tahap kedua adalah *maximum Entropy classifier*. Selain HPS, metode *stemmer* lain yang dibandingkan pada penelitian tersbut adalah *graph based stemmer* (GRAS), YAAS, Linguistica, dan Rule Base *Stemmer*. Bahasa yang digunakan adalah Bahasa Czech, Bahasa Slovakia, Bahasa Poland, Bahasa Hungaria, Bahasa Spanyol, dan Bahasa Inggris. Khusus untuk Polandia dan Slovakia tidak ditemukan metode Rule Base *Stemmer*. Untuk mengetahui pengaruh *stemming* terhadap *Information retrieval*, peneliti menggunakan mean average precision (MAP) dan menggunakan uji statistik *t* dengan *confidence level* 0.95, hasil menunjukkan bahwa metode HPS memberikan hasil yang baik walaupun metode tersebut tidak dirancang khusus untuk kasus *information retrieval* (Brychcín dan Miloslav, 2015).

2.7 Uji Statistik T

Uji statistik t atau t-test adalah metode untuk menguji apakah rata-rata suatu kelompok berbeda dengan rata-rata kelompok yang dibandingkan. T-test merefleksikan perbedaan aktual pada sampel populasi diantara kedua kelompok. Pada metode t-test dilakukan perhitungan *p-value* untuk menemukan perbedaan statistik signifikan pada data yang diuji dengan asumsi awal kedua kelompok data identik. Nilai *p-value<0,05* menunjukkan bahwa data yang diuji berbeda sedangkan nilai *p-value>0,05* menunjukkan bahwa kedua kelompok data yang diuji identik. (Shridan Sriraam, 2017).

1. One sample t-test merupakan teknik analisis untuk membandingkan satu variabel bebas. Teknik ini digunakan untuk menguji apakah nilai tertentu berbeda secara signifikan atau tidak dengan rata-rata sebuah sampel. Pada uji hipotesis ini, diambil satu sampel yang kemudian dianalisis apakah ada perbedaan rata-rata dari sampel tersebut.
2. Analisis Paired-sample t-est merupakan prosedur yang digunakan untuk membandingkan rata-rata dua variabel dalam satu group. Artinya analisis ini berguna untuk melakukan pengujian terhadap satu sampel yang mendapatkan sutau treatment yang kemudian akan dibandingkan rata-rata dari sampel tersebut antara sebelum dan sesudah *treatment*.

2.8 Normalisasi Data

Proses normalisasi data adalah proses untuk mengubah bentuk bahasa yang tidak terstruktur menjadi bahasa Indonesia sesuai dengan ejaan. Proses ini perlu dilakukan karena dapat memberikan peningkatan performa pada performa klasifikasi (Arifiyanti2015). Selain itu, pada data yang dikumpulkan dari media sosial memiliki karakteristik semi-tidak terstruktur. Hal tersebut disebabkan diantaranya karena pengguna media sosial dapat menulis sesuka hati mereka tanpa memperhatikan ejaan bahasa Indonesia yang benar, selain itu dapat disebabkan oleh kesalahan pengguna media sosial dalam menulis kata karena ketidaksengajaan.

Proses normalisasi teks yang dilakukan (Naraadhipa 2011) untuk bahasa Indonesia meliputi empat tahapan, yaitu :

1. Pembersihan angka :

Proses menghapus angka yang ada di awal dan di akhir kata, hal tersebut dilakukan karena angka-angka tersebut tidak berdampak signifikan pada hasil ekstraksi informasi. Contoh : 4malam

2. Penggantian angka :

Proses mengganti angka yang ada di tengah kata. Hal tersebut dilakukan karena ada tren mengganti huruf dengan angka. Contoh : n451b (nasib).

3. Penghapusan karakter berulang :

Proses meminimalkan penggunaan karakter berulang. Contoh : semangaaat (semangat).

4. Proses *translate*:

Proses mengkonversi kata tidak baku menjadi bentuk baku. Dalam proses ini, digunakan bantuan *List* kata tidak baku yang akan di *translate.* (Naraadhipa 2011), (Arifiyanti 2015).

2.9 Transformasi Fitur

Tujuan dilakukannya pembobotan adalah untuk mentransformasi fitur dari data teks menjadi numerik dengan cara menghitung *term* unik. Dalam penelitian ini metode transformasi yang digunakan adalah TF-IDF. Dalam tahap ini, dokumen diwujudkan menjadi sebuah vektor dengan elemen sebanyak *term* yang berhasil dikenali dari tahap ekstraksi dokumen. Vektor tersebut beranggotakan nilai bobot dari setiap *term* yang dihitung berdasarkan metode TF-IDF.

2.9.1 Term Frequency(TF)

Term Frequency (TF) adalah jumlah kemunculan sebuah term pada dokumen. Jika sebuah term sering muncul dalam dokumen, maka *query* yang mengandung term tersebut harus mendapatkan dokumen tersebut.

Metode yang dilakukan untuk mendapatkan nilai TF diantaranya:

1) *Raw* TF

   Pada metode *Raw* TF, nilai *term* didapatkan dengan cara menghitung kemunculan *term* tersebut dalam dokumen.

2) *Logarithmic* TF

   Metode *logarithmic* TF menggunakan fungsi logaritmik

   $tf=1+log(tf)$

   Dimana : *tf* adalah kemunculan kata pada dokumen.

3) *Binary* TF

   Metode ini akan menghasilkan nilai boolean berdasarkan kemunculan *term* pada dokumen tersebut. Bernilai 0 apabila *term* tidak ditemukan, dan bernilai 1 apabila ada. Frekuensi kemunculan *term* pada sebuah dokumen tidak berpengaruh.

4) *Augmented* TF

Metode ini didapatkan dengan rumus:

$Tf = 0{,}5+0{,}5\ max(tf)^{tf}$

Dimana :

tf adalah jumlah kemnculan *term* pada sebuah dokumen.

nilai $max(\text{tf})^{\text{tf}}$ merupakan frekuensi tertinggi kemunculan *term* pada dokumen yang sama.

2.9.2 Metode TF-IDF

Metode TF-IDF ini merupakan metode pembobotan dalam bentuk sebuah metode yang merupakan integrasi antar *term frequency* (TF), dan *inverse document frequency* (IDF) (Fattah, 2015). IDF dihitung dengan rumus :

$$\boldsymbol{IDF_i = \log \frac{D}{DF_i}} \qquad (2.3)$$

Dimana :

IDFi : Inverse Document Frequency dari kata ke-i (ti)

D : jumlah keseluruhan dokumen

DFi : jumlah dokumen yang memiliki kata ti

Fungsi metode ini adalah untuk mencari representasi nilai dari tiap-tiap dokumen dari suatu kumpulan data *training (training set)* dimana nantinya akan dibentuk suatu vektor antara dokumen dengan kata (*documents with terms*) yang kemudian untuk kesamaan antar dokumen dengan *cluster* akan ditentukan oleh sebuah *prototype* vektor yang disebut juga dengan *cluster centroid.*

Nilai TF-IDF meningkat secara proporsional seiring dengan jumlah kata yang muncul dalam dokumen, tetapi diimbangi dengan frekuensi kata yang ada pada korpus, yang membantu untuk mengendalikan fakta bahwa beberapa kata lebih umum dibanding yang lain.

TF-IDF dengan normalisasi frekuensi (TF ternormalisasi) dapat meningkatkan proses pembobtan dari *term*. Fungsi normalisasi untuk mengurangi efek dari panjang dokumen (Sari dkk, 2014).

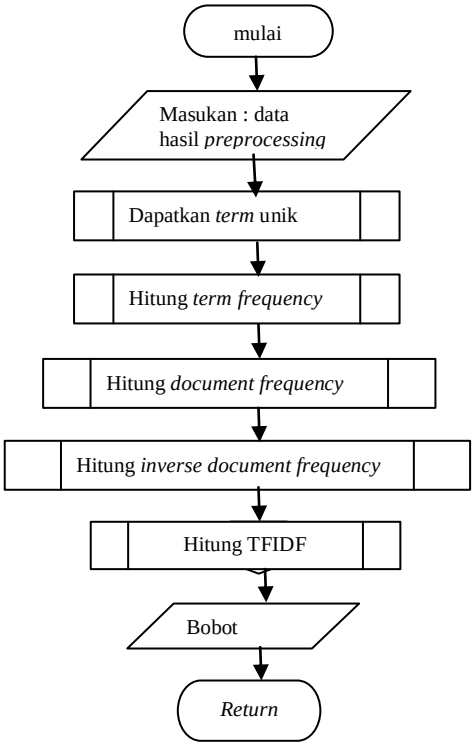
Metode TF-IDF dirumuskan sebagai berikut:

$$\boldsymbol{w(t,d) = tf(t,d) * idf} \quad (2.4)$$

Dimana :

w = bobot.
tf = *term frequency*
idf = *inverse document frequency*

Flowchart pembobotan tf-idf dapat dilihat pada gambar 2.2 berikut,



Gambar 2.2 TF-IDF

2.10 Metode Clustering

Dari sejumlah metode pendekatan *clustering* tersebut terdapat metode *K-means*. Metode *K-means* merupakan salah satu metode non-hierarchical *cluster*ing, metode ini memiliki ketelitian yang cukup tinggi terhadap ukuran objek, sehingga algoritma ini relatif lebih terukurdan efisien untuk pengolahan objek dalam jumlah besar. Selain itu algoritma *K-means* ini tidak terpengaruh terhadap urutan objek. (Ediyanto dkk, 2013).

Selain kelebihan dari metode *K-means*, terdapat juga kelemahan diantaranya pemilihan jumlah klaster yang tepat, pendeteksian *outliers*, dan *cluster overlaping*. pemilihan jumlah klaster yang tepat merupakan masalah laten

dalam metode *K-means*. Beberapa pendekatan telah digunakan dalam menentukan jumlah *cluster* yang paling tepat untuk suatu *dataset* yang dianalisa termasuk di antaranya *Partition Entropy (PE)* dan *GAP Statistics*.Pendeteksian outliers merupakan permasalahan umum yang terjadi hampir di setiap metode yang melakukan pemodelan terhadap data. Khusus untuk metode *K-means* hal ini memang menjadi permasalahan yang cukup menentukan. Beberapa hal yang perlu diperhatikan dalam melakukan pendeteksian *outliers* dalam proses pengelompokan data termasuk bagaimana menentukan apakah suatu data item merupakan *outliers* dari suatu *cluster* tertentu dan apakah data dalam jumlah kecil yang membentuk suatu *cluster* tersendiri dapat dianggap sebagai *outliers.Cluster overlaping* adalah permasalahan menyangkut bentuk *cluster* yang ditemukan. Metode *K-means* umumnya tidak mengindahkan bentuk dari masing-masing *cluster* yang mendasari model yang terbentuk, walaupun secara natural masing-masing *cluster* umumnya berbentuk bundar. (Yudi, 2007).

Metode *K-means* merupakan salah satu metode non-hierarchical *cluster*ing, yang memiliki ketelitian cukup tinggi terhadap ukuran objek, relatif lebih terukur dan efisien untuk pengolahan objek dalam jumlah besar. (Ediyanto dkk, 2013). Langkah-langkah dalam melakukan metode *K-means* adalah :

1. Tentukan jumlah *cluster*
2. Alokasikan data ke dalam *cluster* secara random
3. Hitung *centroid*/rata-rata dari data yang ada di masing-masing *cluster*
4. Alokasikan masing-masing data ke *centroid*/rata-rata terdekat
5. Kembali ke Step 3, apabila masih ada data yang berpindah *cluster* atau apabila perubahan nilai *centroid*, ada yang di atas nilai *threshold* yang ditentukan atau apabila perubahan nilai pada *objective function* yang digunakan di atas nilai *threshold* yangditentukan.

   Untuk menghitung *centroid cluster* ke-*i* menggunakan rumus :

$$\boldsymbol{v} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{\boldsymbol{n}} \qquad (2.5)$$

dimana :

v : centroid pada *cluster*

x_i : objek ke *i*
n : jumlah objek yang menjadi anggota *cluster*.

2.11 Metode Evaluasi

Metode evaluasi dilakukan untuk beberapa macam tujuan. Pada kasus klastering, evaluasi dilakukan diantaranya untuk membandingkan algoritma klastering yang satu dan yang lainnya, membandingkan antara set *cluster*, dan membandingkan antara klaster dengan klaster yang lain. Dalam melakukan evaluasi, dapat dilakukan dengan dua metode yaitu metode evaluasi eksternal dan metode evaluasi internal (Eréndira dkk, 2011).

2.11.1 Evaluasi Eksternal

Evaluasi eksternal dapat dilakukan dengan memanfaatkan label kelas yang telah didefinisikan sebelumnya. Contoh evaluasi menggunakan metode eksternal adalah dengan *Matrix Confusion*, sedangkan parameter yang diperhitungkan adalah *precision, recall,* dan F1-*measure.*

Tabel 2.7 berikut menunjukkan gambaran dari *Matrix Confusion.*(David, 2011).

Tabel 2.7 Matrix Confusion

Kategori X	*Predicted*	
Actual	*True positive*	*False positive*
	False negative	*True negative*

True positive menunjukkan jumlah data uji yang diklasifikasikan sistem kedalam kategori *x,* dan semua data tersebut memang benar merupakan kategori *x.*

False positive menunjukkan jumlah data uji yang tidak diklasifikasikan sistem kedalam kategori *x,* tetapi seharusnya semua data tersebut seharusnya merupakan kategori *x.*

False negative menunjukkan jumlah data uji yang diklasifikasikan sistem kedalam kategori *x,* tetapi seharusnya semua data tersebut bukan merupakan kategori *x.*

True negative menunjukkan jumlah data uji yang tidak diklasifikasikan sistem kedalam kategori *x,* dan semua data tersebut memang bukan merupakan kategori *x.*

Untuk parameter yang digunakan untuk mengukur performa adalah *precision*. *Precision* dihitung dengan rumus 2.9 berikut.

Precision = *True positive* / (*True positive* + *False positive*) (2.9)

Precision adalah keakuratan hasil klasifikasi sistem, sehingga dapat diketahui apakah kategori data yang diklasifikasi oleh sistem sesuai dengan kategori sebenarnya. (David, 2011).

Parameter yang kedua adalah *Recall.* Nilai *Recall* menunjukkan tingkat keberhasilan sistem dalam mengenali suatu kategori. Nilai *Reecall* dihitung dengan menggunakan persamaan 2.10 berikut. (David, 2011)

Recall = *True positive* / (*True positive* + *False negative*) (2.10)

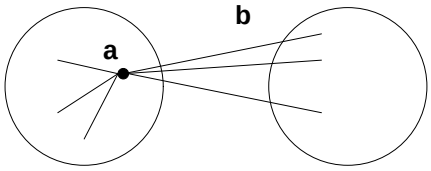
Parameter yang ketiga adalah F1-*measure.* F1-*measure* adalah gambaran pengaruh relative antara *precision* dan *recall* , sehingga performa metode yang digunakan dapat disimpulkan dari nilai F1-*measure.* Nilai F1-*measure* dihitung dengan menggunakan persamaan 2.11 berikut. (David, 2011)

F1-*measure* = (2**Precision*)**Recall* / (*Precision* + *Recall*) (2.11)

2.11.2 Evaluasi Internal

Evaluasi internal dilakukan apabila tidak ada label kelas yang digunakan sebagai pembantu dalam mengevaluasi hasil algoritma *clustering*. Evaluasi Internal dilakukan dengan memperhitungkan korelasi dan similiaritas *cluster* dalam data. Salah satu contoh evaluasi Internal adalah dengan metode *silhouette index* (Rendóndkk, 2011). *Silhouette index* menunjukkan seberapa besar kedekatan objek dalam satu *cluster*dan seberapa jauh jarak antar satu *cluster*

dengan *cluster* yang lain. Untuk memperjelas pemahaman *silhouette index* dapat dilihat pada Gambar 2.4 berikut.



Gambar 2.4*Silhouette*

dimana :

a= rata-rata jarak titik *i* terhadap setiap objek pada *cluster* yang sama.

b= rata-rata jarak titik *i* terhadap setiap objek pada *cluster* yang lain.

Silhouette index didefinisikan dengan persamaan (2.6)

$$(i) = \frac{(b(i)-a(i))}{Max\{a(i),b(i)\}} \tag{2.6}$$

dimana :

i = Silhouette index

a(i)= rata-rata jarak objek *i* dengan seluruh objek pada *cluster* tempat asal *i*

b(i)=nilai minimum dari rata-rata jarak objek *i* dengan seluruh objek pada *cluster* yang lain.

Calinski-Harabaz index, merupakan metode yang mirip *silhouette* untuk mengevaluasi jarak *internal cluster* dan *external cluster.Calinski-Harabaz index* didefinisikan dengan persamaan 2.12 berikut.

$$CH = \frac{trace(S_B)}{trace\ (S_w)} * \frac{n_{p-1}}{n_p - k} \tag{2.12}$$

dimana :

$trace(S_B)$: matriks jarak antar *cluster.*

$trace\ (S_w)$: matriks jarak internal *cluster.*

n_p : jumlah sampel yang terklaster.

k :jumlah *cluster.*

Davies-Bouldin index, merupakan metode evaluasi *cluster* internal yang bertujuan untuk mengidentifikasi kelompok-kelompok *cluster* yang rapat dan

terpisah dengan baik dengan *cluster* lainnya (Rendóndkk, 2011). *Davies-Bouldin index* didefinisikan dengan persamaan 2.13 berikut.

$$DB = \frac{1}{c}\sum_{i=1}^{c} Max_{i \neq j}\left\{\frac{d(X_i)+d(X_j)}{d(c_i,c_j)}\right\} \quad (2.13)$$

dimana :

c= jumlah *cluster*

i,j= label *cluster*

$d(X_i)d(X_j)$: jarak sampel-sampel pada *cluster i* dan *j* terhadap pusat *cluster* masing-masing.

$d(c_i, c_j)$: jarak antara pusat *cluster i* dan pusat *cluster j.*

Nilai *DB* yang lebih kecil diindikasikan sebagai solusi *cluster* terbaik (Rendóndkk, 2011).

2.11.3 Evaluasi Metode *Stemmer*

Metode untuk mengevaluasi *stemmer* dapat menggunakan beberapa cara, diantaranya menggunakan indeks *overstemming*, indeks *understemming*, *stemming weight*, dan *error rate to trunctuation*(Paice, 1996). Daftar kelompok-kelompok kata-kata berimbuhan yang memiliki kesamaan kata dasar dibutuhkan untuk melakukan perhitungan indeks *overstemming* dan indeks *understemming.* Dari kelompok kata tersebut kemudian dihitung nilai *desired merge total* (DMT) dan *desired non-merge total* (DNT) untuk setiap kelompok kata berimbuhan sebagai berikut :

$$DMT = 0{,}5n(n-1) \quad (2.14)$$

dimana

DMT : *desired merge total*

n : jumlah kata pada suatu kelompok

$$DNT = 0{,}5n(W-n) \quad (2.15)$$

dimana

DNT : *desired non-merge total*

n : jumlah kata pada suatu kelompok

W : jumlah kelompok kata

Nilai *global wrongly-merged total* (GWMT) dan *global desirednon-merge total* (GDNT) kemudian dihitung dengan menjumlah seluruh nilai DMT dan

DNT. Kemudian dilakukan perhitungan untuk mencari indeks *overstemming* dan indeks *understemming*:

1) *Overstemming* terjadi ketika metode *stemmer* salah dalam menghapus imbuhan dari kata dasar, dimana kata dasar ikut terhapus sebagian. Kesalahan tersebut dapat menyebabkan kata yang tidak berelasi ter-*stemming* menjadi satu kata dasar.Untuk mendapatkan nilai indeks *overstemming*, pertama dengan melakukan *stemming* pada daftar kelompok-kelompok kata-kata berimbuhan.Kemudian dilakukan pencarian dimana ada kata-kata dasar yang sama, namun berasal dari kelompok yang berbeda dan disebut sebagai *wrongly-merged total* (WMT). WMT dihitung dengan cara:

$$WMT = 0{,}5 \sum v_i (n_s - v_i) \qquad (2.16)$$

dimana
WMT : *wrongly-merged total*
$\sum_{vi}$: jumlah kata yang berasal dari kelompok berbeda
n_s : jumlah kata pada kelompok tersebut

Dengan menjumlah seluruh nilai WMT maka didapatkan nilai *global wrongly-merged total* (GWMT). Indeks *overstemming* kemudian didapatkan dari hasil pembagian GWMT/GDNT.

2) *Understemming* terjadi saat metode *stemmer* salah dalam menghapus imbuhan dari kata dasar, dimana imbuhan masih tertinggal pada kata dasar atau hanya sebagian imbuhan yang terhapus. Kesalahan tersebut dapat menyebabkan kata yang memiliki kata dasar sama ter-*stemming* menjadi kata dasar yang berbeda.Nilai indeks *understemming* didapatkan setelah melakukan *stemming* pada daftar kelompok-kelompok kata-kata berimbuhan. Kemudian dilakukan penghitungan ada berapa kata dalam satu kelompok yang tidak ter-*stemming* menjadi kata dasar yang sama. Maka pada kelompok tersebut dapat dihitung nilai *unachieved-merge total* (UMT). UMT dihitung dengan cara :

$$UMT = 0{,}5 \sum u_i (n_s - u_i) \qquad (2.17)$$

dimana :
UMT : *unachieved-merge total*
$\sum_{ui}$: jumlah kata yang tidak ter-*stemming* menjadi kata dasar yang sama pada kelompok tersebut
n_s : jumlah kata pada kelompok tersebut

Dengan menjumlah seluruh nilai UMT maka didapatkan nilai *global unachieved-merge total* (GUMT). Indeks *understemming*kemudian didapatkan dari hasil pembagian GUMT/GDMT.

3) *Stemming weight* merupakan rasio perbandingan antara indeks *overstemming*dan indeks *understemming*.*Stemming weight*kemudian didapatkan dari hasil pembagian *overstemming* /*understemming*.

2.12 Metode Pemilihan Untuk Memilih *Stemmer*

Pada sub-bab ini akan dibahas mengenai metode pemilihan yang akan digunakan untuk menentukan metode *stemmer* terbaik. Metode *stemmer* dipilih dengan mempertimbangkan hasil pengujian yaitu pengujian performa metode *stemmer* dan hasil pengujian performa *clustering.* Performa metode *stemmer* meliputi indeks *understemming* dan *overstemming.* Sebagai tambahan, waktu pemrosesan metode *stemmer* juga ikut dipertimbangkan. Sedangkan untuk performa *clustering,* digunakan nilai DBI.

Metode pemilihan yang digunakan adalah *weighted score.* Selain itu, justifikasi pemberian nilai bobot juga akan dibahas pada sub-bab ini.

2.12.1 Metode Weighted Scoring

Metode *weighted scoring* dapat digunakan untuk membatu melakukan pemilihan dari beberapa kriteria. Metode tersebut telah digunakan secara luas untuk melakukan evaluasi dan membantu para pembuat keputusan untuk menentukan pilihan (Jadhav dan Sonar, 2009). Metode *weighted score* hanya dapat diterapkan pada data numerik sehingga cocok digunakan pada indeks *understemming, overstemming,* maupun nilai DBI.

Diasumsikan terdapat alternatif pilihan sejumlah $m\{A_1, A_2, \ldots, A_m\}$ dengan kriteria $n\{C_1, C_2, \ldots, C_n\}$, maka alternatif ditentukan dengan matriks pemilihan $\{S_{ij}\}$, dimana S_{ij} adalah skor yang menunjukkan seberapa baik alternatif A pada kriteria C. Bobot atau *weight* $\{W_1, W_2, \ldots, W_k\}$ menunjukkan tingkat

kepentingan dari kriteria C. Alternatif terbaik ditentukan dari nilai skor terbesar (Jadhav dan Sonar, 2009).

Tabel 2.8 berikut menunjukkan contoh perhitungan *weighted scoring* pada pemilihan *software component* (Jadhav dan Sonar, 2009).

Tabel 2.8 *Weighted Scoring* Pada Pemilihan *Software Component* (Jadhav dan Sonar, 2009)

Component	Criteria	Weight	Rating	Score
1	User Satisfaction	35	5	175
	Service	35	4	140
	Access Control	10	1	10
	Error Prone	20	5	100
total				425

2.12.2 Justifikasi Pemberian Nilai Bobot Pada Metode *Weighted Scoring* Untuk Pemilihan Metode *Stemmer.*

Untuk menentukan nilai bobot pada metode *weighted scoring* dibutuhkan dasar dan alasan yang mendukung. Oleh sebab itu, pada sub-bab ini akan dibahas literatur yang dijadikan dasar penentuan nilai bobot. Kriteria yang digunakan pada penelitian ini adalah performa metode *stemmer* yang meliputi indeks *understemming,* indeks *overstemming,* dan waktu proses. Dampak pada *information retrieval* diukur dari performa hasil *clustering* menggunakan nilai DBI. Literatur yang digunakan sebagai pertimbangan untuk menentukan nilai bobot akan disajikan pada Tabel 2.9 berikut.

Tabel 2.9 Penelitian Tentang Metode *Stemming* Dalam Berbagai Bahasa

Nomor	Sumber	Ringkasan
1	Jaafar dkk, 2017	Penelitian tersebut membahas tentang perbandingan metode-metode stemmer bahasa Arab dan mengusulkan metode SAFAR-Stemmer. Performa metode stemmer diukur dengan*Accuracy, Index Compresion Factor,* dan *G-score.* Waktu pemrosesan juga dipertimbang-kan (*Execution Time*). Metode stemmer yang digunakan yaitu *Light10, Motaz, Tasaphyne,* dan *SAFAR.* Kesimpulan yang didapat, metode *SAFAR* mendapat-

Tabel 2.9 Penelitian Tentang Metode *Stemming* Berbagai Bahasa (Lanjutan)

Nomor	Sumber	Ringkasan
		-akurasi terbesar yaitu 33% dan *G-score* pada posisi kedua yaitu 0,1. Namun demikian, walaupun tujuan *stemmer* untuk diterapkan pada kasus *information retrieval*, metode *SAFAR* belum diterapkan pada kasus seperti klasifikasi atau pengelompokan data.
2	Flores dan Moreira, 2016	Seperti yang telah dibahas pada Bab 1, penelitian dilakukan dengan membandingkan berbagai metode *stemmer* pada empat bahasa. Pada penelitian tersebut, performa metode *stemmer* diukur dengan metode *paice*. Performa pada *information retrieval* diukur menggunakan *mean average precision* (MAP). Sedangkan waktu proses metode *stemmer* tidak dibahas.
3	Shrestha dan Dhakal, 2016	Peneliti mengajukan metode *stemmer* baru untuk bahasa Nepali. Bahasa Nepali adalah bahasa Indo-Arayan yang ditulis dengan teks Devanagari. Metode *stemmer* tersebut termasuk *rule based* dengan total 128 aturan, dan diuji pada 5000 kata Nepali. Performa metode *stemmer* diukur menggunakan *accuracy*. Rata-rata akurasi yang didapat adalah 88,78%. Untuk waktu proses metode *stemmer* tidak disebutkan.
4	Dalwadi dan Desai, 2016	Peneliti mengajukan metode *stemmer* baru untuk bahasa Gujarati. Sama seperti bahasa Nepali, bahasa Gujarati, Hindi, Bengali, Marathi, termasuk dalam rumpun bahasa Indo-Arayan. Berbeda dengan metode usulan Shrestha dan Dhakal (2016), metode *stemmer* ini menggunakan kamus untuk melakukan proses *stemming*.Metode ini termasuk jenis *rule based* dengan total 227 aturan.Performa *stemmer* diukur menggunakan *accuracy*, 96.63% pada 6530 kata, sedangkan waktu proses tidak dibahas.

Tabel 2.9 Penelitian Tentang Metode *Stemming* Berbagai Bahasa (Lanjutan)

Nomor	Sumber	Ringkasan
5	Boukhari dan Omri, 2015	Peneliti mengajukan metode *stemmer* baru untuk bahasa Inggris yang merupakan perbaikan dari metode sebelumnya (porter dan lovin). Metode yang diberi nama SAID tersebut memiliki tahap pengecekan *compound words* dan tambahan aturan pada penghapusan *suffixes*. Metode diuji denganperbandingan terhadap metode porter dan lovin. Performa *stemmer* dihitung berdasarkan *recall* dan *precision*. Performa *stemmer* pada *information retrieval* belum diuji, sedangkan waktu pemrosesan juga tidak disebutkan.
6	Brychin dan Konopik, 2015	Sama seperti Flores dan Moreira (2016), pada penelitian ini juga dibandingkan beberapa metode stemmer pada enam bahasa. Namun peneliti mengusulkan metode yang disebut *high precision stemmer* (HPS). Untuk mengukur performa *stemmer* tidak menggunakan metode *paice*, tetapi dengan *recall*, *precision,*dan *f-measure*. Performa pada *information retrieval* diukur menggunakan *mean average precision* (MAP). Waktu pemrosesan metode *stemmer* tidak dibahas pada penelitian tersebut.
7	Meitei dkk, 2015	Para peneliti mengajukan metode *stemmer* untuk bahasa Manipur yang dikembangkan berdasarkan metode porter (1980) dengan menghapus imbuhan. Perbedaan dengan metode porter, pada *stemmer* baru tersebut digunakan proses pengecekan kata pada *database*. Untuk menguji performa metode *stemmer*, digunakan *accuracy* dan *average accuracy*. Metode *stemmer* belum dicobakan pada kasus *information retrieval*. Sedangkan waktu pemrosesan data oleh *stemmer* tidak disebutkan.

Tabel 2.9 Penelitian Tentang Metode *Stemming* Berbagai Bahasa (Lanjutan)

Nomor	Sumber	Ringkasan
8	Mahmud dkk, 2014	Para peneliti mengajukan dua metode *stemmer* untuk bahasa Bengali. Seperti metode porter, metode *stemmer* ini tidak mengecek kamus dalam melakukan proses *stemming*. Metode pertama digunakan untuk *verbal inflected words*, sedangkan yang kedua untuk *noun inflected words*. Total kata yang digunakan untuk pengujian adalah 4500 kata. Performa metode *stemmer* diukur dengan nilai *accuracy*. Waktu pemrosesan danmemori sistem pada saat metode dijalankan juga dihitung. Hasil pengujian menunjukkan akurasi 83% untuk *verb* dan 88% untuk *noun*.
9	Gupta dkk, 2013	Para peneliti mengajukan metode *stemmer* untuk bahasa Urdu. Urdu merupakan bahasa kombinasi dari bahasa Arab, Hindi, Turki, Sanskrit dan lainnya, sehingga terdapat banyak kemiripan dengan bahasa rumpun Indo-Arayan. Metode ini juga tidak menggunakan kamus, dan terdiri dari total 238 aturan. Pengujian performa *stemmer* dilakukan dengan menghitung *accuracy*. Data yang digunakan berjumlah 2000 kata, dan metode tersebut mendapat nilai akurasi 86,5%. Untuk waktu proses metode tidak dibahas pada penelitian tersebut.

Dari pembahasan literatur yang telah dilakukan pada Tabel 2.9, tahap selanjutnya adalah melakukan pembobotan kriteria untuk setiap metode *stemmer*. Tahap pembobotan kriteria akan disajikan pada Tabel 2.10. Langkah-langkah yang dilakukan adalah mengumpulkan justifikasi dari penelitian-penelitian terdahulu, kemudian dilakukan pengambilan keputusan mengenai besarnya nilai bobot yang diberikan pada kriteria metode *stemmer*.

Tabel 2.10 Justifikasi Pemberian Nilai Bobot

Nomor	Kriteria	Bobot	Justifikasi Pemberian Bobot
1	Nilai DBI	45%	• *stemmers* biasanya digunakan pada proyek-proyek pengembangan aplikasi, terutama pada pengembangan *information retrieval system* (Jaafar, 2017). • *stemmers* digunakan secara luas pada *text mining, information retrieval*, dan *natural language processing systems* untuk meningkatkan performa sistem-sistem tersebut (Shrestha 2016). • sistem yang dikembangkan (*affix removal stemmer for Gujarati text*) dapat bermanfaat untuk aplikasi *information retrieval, dictionary search, document summarization, machine translation* (Dalwadi 2016). • *stemmers* pada dasarnya digunakan pada *information retrieval system* untuk meningkatkan performa (Meitei, 2015). • *stemmers* digunakan pada *search engines, word processing,spell checkers* (Gupta, 2013).
2	Nilai *Understemming*	20%	• pada penelitian oleh Flores dan Moreira (2016), dilakukan pengujian korelasi antara *understemming, overstemming,* dengan *mean average precision* (MAP) menggunakan metode *pearson*. Hasil menunjukkan bahwa *overstemming* berkorelasi negatif signifikan (-0,70) untuk bahasa Inggris, (-0,91) untuk bahasa Perancis, dan (-0,73) untuk bahasa Spanyol. Sedangkan pada ketiga bahasa tersebut, nilai *understemming* berkorelasi positif tidak signifikan, yaitu (0,42), (0,33), dan (0,29). Dapat disimpulkan bahwa *overstemming error* lebih serius dari *understemming error*.
3	Nilai *Overstemming*	25%	

Tabel 2.10 Justifikasi Pemberian Nilai Bobot (Lanjutan)

Nomor	Kriteria	Bobot	Justifikasi Pemberian Bobot
4	Waktu Proses	10%	• Penelitian yang membahas waktu proses dan kompresi data dilakukan oleh Jaafar dkk (2017), sedangkan yang membahas waktu proses dan memori sistem dilakukan oleh Mahmud dkk (2014). Penelitian lain pada Tabel 2.9 tidak membahas waktu proses.

2.13 Metode Normalisasi Nilai

Kriteria yang digunakan untuk penilaian pada metode *weighted scoring* memiliki skala yang berbeda. Contohnya bagaimana membandingkan nilai DBI pada metode *nazief stemmer* dengan waktu proses yang memiliki satuan detik. Agar dapat melakukan perbandingan, harus dilakukan eliminasi dari unit pengukuran. Operasi tersebut dikenal dengan sebutan *normalize* (Abdi, 2010). Selain itu, terdapat beberapa istilah serupa seperti *feature scaling, feature normalization,unity-based normalization,* dan *scale normalization.*

Pada penelitian yang dilakukan oleh Alam, dkk (2011) melakukan perbandingan teknik-teknik normalisasi fitur untuk *speaker verification system. Speaker verification* merupakan tahap untuk mengenali dan memverifikasi data suara. Pada penelitian tersebut dilakukan tahap *scale normalization* :

$$X_{sn} = \frac{x - b_i}{b_u - b_i} \quad (2.18)$$

Xsn adalah nilai ternormalisasi yang memiliki rentang diantara 0 sampai dengan 1. Sedangkan *x* adalah komponen fitur, b_u adalah batas tertinggi, dan b_i adalah batas terendah.

Pada penelitian ini, kriteria metode *stemmer* yaitu nilai DBI, nilai *understemming,* nilai *overstemming,* dan waktu proses terendah harus memiliki skor tertinggi. Oleh sebab itu perlu dilakukan modifikasi dari Persamaan 2.18 sebagai berikut :

$$X_a = \frac{x_{max} - x}{x_{max} - x_{min}} \quad (2.19)$$

Diasumsikan terdapat metode *stemmer a, b,* dan *c* serta kriteria I, II, dan III. Maka, X_a adalah nilai kriteria I dari metode *stemmer a* yang telah dinormalisasi, *x* adalah nilai kriteria I dari metode *stemmer a*, x_{max} adalah nilai tertinggi pada kriteria I, dan x_{min} adalah nilai terendah pada kriteria I.

2.14 Kontribusi Penelitian

Penelitian ini memberikan kontribusi yaitu pada perbandingan metode-metode *stemmer*bahasa Indonesia. Hasil penelitian ini akan menunjukkan metode yang paling singnifikan untuk penggalian teks pada bahasa Indonesia. Selain itu, akan dikonfirmasi apakah pola yang ditemukan oleh Flores dan Moreira (2016) akan berlaku pada bahasa Indonesia. Penulis juga melakukan perbaikan aturan pada metode *arifiyanti* (2015) sebagai tambahan kontribusi keilmuan.

Untuk kontribusi praktis, penulis melakukan pemilihan metode *stemmer* dengan menggunakan metode *weighted scoring.* Metode yang terpilih menjadi saran untuk implementasi penanganan keluhan pelanggan PLN dari data *twitter*. Selain itu, penulis merancang desain sistem penanganan keluhan pelanggan untuk PT.PLN.

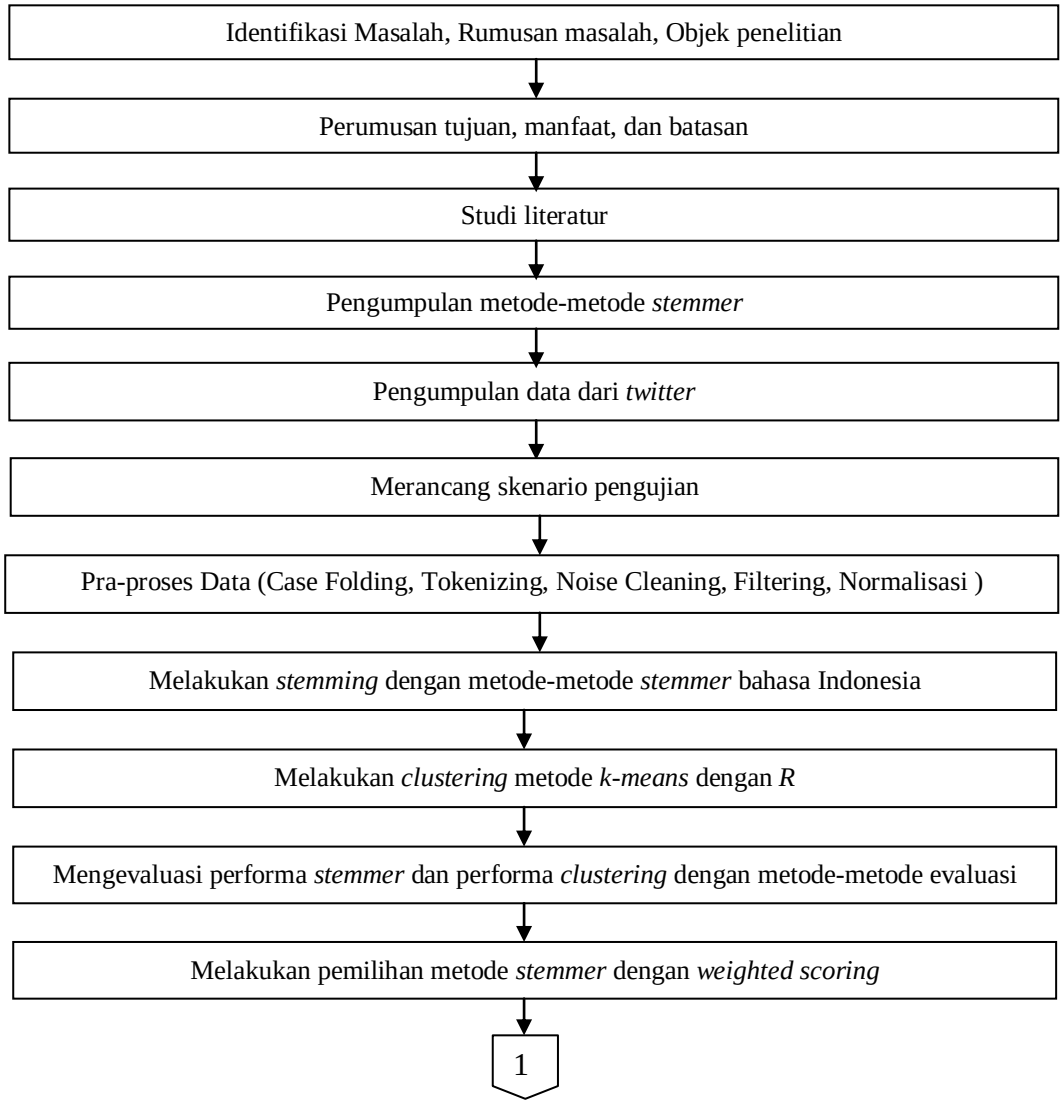
Halaman ini sengaja dikosongkan

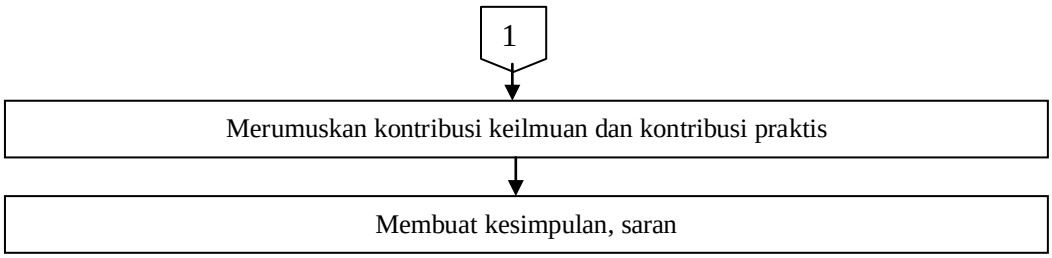
BAB 3
METODE PENELITIAN

Pada bab ini akan dijelaskan tentang metodepenelitian untuk mengektraksi fitur pada keluhan pelanggan PLN, pada bab ini akan dijabarkan tentang langkah-langkah pelaksanaan penelitian, termasuk metode pengumpulan data, metode dalam melakukan pra-proses data, serta analisis hasil penelitian.

3.1 Gambaran Umum Penelitian

Langkah-langkah pelaksanaan penelitian untuk tesis ini dapat dilihat pada Gambar 3.1 tentang gambaran umum penelitian, yaitu seperti dibawah ini.





Gambar 3.1 Gambaran Umum Penelitian

Langkah pertama dalam penelitian ini adalah melakukan identifikasi masalah, perumusan tujuan, manfaat, dan batasan penelitian, sebagaimana telah dipaparkan pada Bab I. Setelah itu dilakukan studi literatur tentang penelitian terdahulu dan dasar teori yang akan menunjang penulisan tesis. Teori-teori yang dibutuhkan adalah pemahaman tentang metode *text mining,* meliputi ekstraksi fitur, serta pemahaman tentang metode *clustering k-means*. Literatur yang digunakan diambil dari jurnal penelitian dan buku. Bagian ini telah dibahas pada Bab II.

Kemudian dilakukan pengumpulan dataset yang dibutuhkan dalam penelitian berupa komentar atau *tweet* pada media sosial twitter yang berasal dari pelanggan PLN.Dari data set yang terkumpul, disusun data kelompok kata yang akan dijelaskan pada Sub-bab 3.2

Langkah ini diteruskan dengan merancang skenario pengujian. Selanjutnya, dilakukan pra-proses data untuk digunakan dalam pegujian metode-metode stemmer.

Setelah dilakukan proses *stemming* pada data, dilakukan tahap transformasi fitur. Tahap ini digunakan untuk mendapatkan nilai dari kata yang berhasil diekstrak dengan cara mengubah kata-kata tersebut kedalam bentuk numerik. Metode yang digunakan adalah *term frequency-invers document frequency.* Contoh transformasi fitur dengan metode TF-IDF dapat dilihat pada Tabel 3.1 berikut.

Tabel 3.1Transformasi Fitur DenganMetode TF-IDF

		Term frequency pada dokumen														
no	*term*	a	b	c	d	e	f	g	h	i	j	k	l	x	DF	IDF
1	anggota	1	0	0	0	0	0	0	0	0	0	0	0	0	**1**	1.079
2	abiotik	0	0	0	0	0	0	0	0	0	2	1	0	0	**2**	0.778

Data yang telah ditransformasi dengan metode TF-IDFmenjadi matriks, dilakukan *clustering.*Pada proses ini dilakukan pengelompokan data dengan metode *k-means* menggunakan perangkat lunak pembantu *R Studio*. *R Studio* dan bahasa pemrograman *R* adalah paket perangkat lunak yang dikenal luas, serta sering digunakan untuk mengimplementasikan algoritma *data mining,* metode-metode statistik, dan pengolahan grafis.

Data hasil pra-proses diubah menjadi data dengan format *csv* dan dijadikan masukan pada *R Studio*. Langkah-langkah yang dilakukan yaitu mengaktifkan *library* yang dibutuhkan untuk pemrosesan data teks pada *R Studio,* mendefinisikan lokasi data filedan nama file yang diinginkan, mentransformasi data menjadi matriks, menjalankan fungsi *tf-idf,* dan yang terakhir adalah menjalankan fungsi *k-means.*

Pada bagian memilih *clusterer,* perlu didefinisikan nilai *k* Nilai *k* menunjukkan jumlah kelompok data yang diinginkan. terdapat beberapa metode untuk menentukan nilai *k* yang diinginkan yaitu *Gaussian-means*, *Elbow method, cross validation method, An Information Theoretic Approach,* dan *Silhoutte method, rule of thumb* (Kodinariya dkk, 2013). Pada penelitian ini digunakan beberapa nilai *k,* dimulai dengan *k=2* sampai dengan 20.

Setelah proses *clustering* dan *stemming* pada data kelompok kata, dilakukan evaluasi yang meliputi evaluasi performa stemmer menggunakan indeks *overstemming,* dan indeks *understemming.* Sedangkan untuk mengevaluasi *cluster* digunakan *davies-bouldin index* (DBI). Untuk menghitung nilai DBI pada *R Studio* dibutuhkan *library clValid.*Uji *t* dilakukan untuk mengetahui signifikansi metode *stemmer*.

Tahap selanjutnya adalah melakukan pemilihan metode *stemmer* terbaik dengan metode *weighted scoring* serta merumuskan kontribusi penelitian.

3.2 Pengumpulan Data

Data yang diperlukan dalam penelitian ini adalah data komentar pelanggan PLN yang didapatkan dari akun resmi PLN yaitu pln123. Data yang dikumpulkan dimulai dari bulan Agustus sampai September 2016. Metode pengumpulan data yang digunakan adalah dengan menggunakan bantuan twitter API dan perangkat lunak pembantu *R*. Untuk *library* twitter menggunakan *library* twitter *open source*. Data yang dikumpulkan tersebut disimpan dalam format *csv*. Contoh data yang terkumpul sebagai berikut.

@pln_123 maksudnya pak untuk daerah sumatera utara ada gak mati lampu?
@pln_123 daerah wolter monginsidi listrik mati..knp ya? Lama ga?
@pln_123 palembang kecamatan ib1 padam boss.. tolong ditindaklanjuti
@pln_123 ,lampu jalan hampir 1 tahun tidak menyala bos, kan setiap bayar listrik ada Biaya PPJ, apakah Biaya PPJ trmasuk utk lampu jlan ?
@pln_123 di kalimantan tengah krisis meteran listrik. Mohon segera ditindak lanjuti,krisis meteran sudah 1 tahun
@PLN_TGK @pln_123 @pringsewuupdate Padam listrik jam 18:10 td di Jl.Gholib RT.1, Kel. Pringsewu Barat, Kab.Pringsewu-LAMPUNG. #lampunggelap
@pln_123 menyala gan, pukul 21:30. Makasiii PLN
@pln_123 informasi utk padam listrik di Wilayah Pekanbaru ada gk Min? Udah 2hari listrik padam tanpa ada pemberitahuan
@pln_123 gimana sih mati listrik 2 kali , bikin rusak peralatan elektronik. harga udah naik kualitas makin buruk. payah
@pln_123 kpn nyalanya pak

(Twitter @pln_123)

Selain menyiapkan data untuk *clustering*, juga disiapkan data untuk melakukan evaluasi performa metode-metode *stemmer* dengan metode *paice* (1980). Data yang digunakan untuk pengujian performa metode *stemmer* berupa kelompok-kelompok kata sebagai berikut :

Tabel 3.2 Contoh Data Kelompok Kata

Kelompok	**Kata**
1	akhir akhirnya pengakhiran diakhiri
2	balas dibalas pembalasannya terbalaskan membalaskan
3	cepat secepatnya kecepatan dipercepat
4	dapat mendapat mendapati pendapatannya didapatinya
5	ganti mengganti pergantian menggantikan tergantikan

3.3 Skenario Pengujian

Ada tiga skenario pengujian yang digunakan, yaitu pengujian performa *stemmer*, performa *clustering*, dan signifikansi hasil *clustering*.

Pengujian I

Skenario pengujian pertama dilakukan untuk menguji performa metode *stemmer* yang digunakan. Nilai *Understemming* menunjukkan performa metode *stemmer* dalam mendapatkan kata dasar dari kata-kata yang memiliki kesamaan makna. Nilai *Overstemming* menunjukkan performa metode *stemmer* dalam mendapatkan kata dasar dari kata-kata yang memiliki perbedaan makna.

Tabel 3.3 Pengujian I

nomor	Nama Metode *Stemmer*	Indeks *Understemming*	Indeks *Overstemming*	*Stemming weight*
1	*Porter confix striping stemmer*			
2	*Nazief Stemmer*			
3	*Arifin Stemmer*			
4	*Fadillah Stemmer*			
5	*Asian Stemmer*			
6	*Enhanced Confix Stripping Stemmer*			
7	*Arifiyanti Stemmer*			

Pengujian II

Skenario pengujian yang kedua dilakukan untuk mengetahui pengaruh metode-metode *stemmer* yang digunakan terhadap *Information retrieval*. Pada Tabel 3.4 berikut berisi informasi tentang jumlah kelas dan nilai DBI dari setiap metode *stemmer* yang diterapkan.

Tabel 3.4 Pengujian II

Jumlah Kelas	***Porter confix stripping***	***Nazief***	***Ari-fin***	***Fadil-lah***	***Asian***	***Enhanced Confix Stripping***	***Arifi-yanti***	***Tanpa Stemmer***
2								
...								

Pengujian III

Skenario pengujian yang ketiga dilakukan untuk mengetahui tingkat signifikan setiap metode *stemmer* terhadap data tanpa dilakukan proses *stemming*. Pada pengujian kedua ini dilakukan uji *t* statistik. Pada tabel tersebut akan digunakan simbol "-" yang menunjukkan metode *stemmer* pada baris kurang dari metode *stemmer* pada kolom, simbol "+" yang menunjukkan metode *stemmer* pada baris lebih baik dari metode *stemmer* pada kolom, dan simbol "=" yang menunjukkan metode *stemmer* pada baris sama dengan metode *stemmer* pada kolom (Brychcín dan Konopík, 2015).

Tabel 3.5 Pengujian II

	Tanpa Stemmer	***Arifiyanti***	***Enhanced Confix Stripping***	***Asian***	***Fadillah***	***Arifin***	***Nazief***	***Porter confix stripping***
Tanpa Stemmer								
Arifiyanti								
Enhanced Confix Stripping								
Asian								
Fadillah								
Arifin								
Nazief								

3.4 Pra-proses Data

Tahap pra-proses data dilakukan untuk menyiapkan data agar siap digunakan pada proses selanjutnya. Selain itu, pra-proses dilakukan dengan tujuan untuk mengurangi beban sistem secara keseluruhan. Tahap ini terdiri dari *case folding, tokenizing, filtering, dan stemming.*

3.4.1 Case Folding

Tahap *case folding* terdiri dari proses pengubahan karakter kapital menjadi non-kapital dan proses penggantian karakter selain huruf yang disebut *delimiter* menjadi spasi (“ “).Contoh *case folding* pada Tabel 3.6 berikut.

Tabel 3.6Case Folding

Masukan	Hasil
Padam listrik jam 18:10 tadi di Jalan.Gholib RT.1, Kel. Pringsewu Barat, Kab.Pringsewu-LAMPUNG.	padam listrik jam 18 10 tadi di jalan gholib rt 1 kelpringsewu barat kab pringsewu lampung

3.4.2 Tokenizing

Tahap *tokenizing* dilakukan untuk memecah dokumen menjadi *token-token* berdasarkan pembatas spasi (“ “). Keluaran dari proses ini adalah token-token yang tersimpan dalam bentuk *array.* Contoh *tokenizing* pada Tabel 3.7 berikut.

Tabel 3.7Tokenizing

Masukan	Hasil
@pln_123 di kalimantan tengah krisis meteran listrik. Mohon segera ditindak lanjuti, krisis meteran sudah 1 tahun	di | kalimantan | tengah | krisis | meteran | listrik. | Mohon | segera | ditindak | lanjuti, | krisis | meteran | sudah | 1 | tahun |

3.4.3 Noise Cleaning

Tahap *noise cleaning* dilakukan untuk menghapus data yang tidak digunakan dalam penelitian. Hal ini disebabkan karena data yang diambil dari media sosial memiliki beberapa jenis. Yang pertama adalah data *tweet,* yaitu tulisan yang dikirimkan kepada pihak @pln_123. Yang kedua adalah data *retweet,* yang merupakan pengiriman ulang oleh pengguna twitter. Pada proses ini data *retweet* dihapus, selain itu nama pengguna *tweet* juga dihapus. Contoh nama pengguna *tweet* adalah “@pln_123”. Sedangkan contoh *noise cleaning* pada Tabel 3.8 berikut.

Tabel 3.8Noise Cleaning

Masukan	Hasil
@pln_123 di kalimantan tengah krisis meteran listrik. Mohon segera ditindak lanjuti, krisis meteran sudah 1 tahun	di kalimantan tengah krisis meteran listrik. Mohon segera ditindak lanjuti, krisis meteran sudah 1 tahun

3.4.4 Filtering

Tahap filtering adalah tahap pengambilan kata-kata penting yang akan digunakan untuk tahap selanjutnya. Metode yang digunakan dalam proses ini adalah metode *stopwords removal* yaitu dengan membuang kata-kata yang dianggap tidak penting. Daftar *stopwords* yang digunakan berasal dari penelitian yang dilakukan oleh Fadillah (Tala, 2003).

3.4.5 Penghapusan Karakter Berulang

Pada tahap ini dilakukan proses normalisasi data yang berupa penghapusan karakter berulang. Teknik normalisasi data menggunakan metode yang diusulkan oleh (Aqsath dan Ayu, 2011). Proses penghapusan karakter berulang dilakukan sebagai upaya untuk dapat memaksimalkan kinerja metode-metode *stemmer*. Hal tersebut disebabkan karena metode *stemmer* berbasis aturan akan melakukan pencocokkan karakter atau huruf pada kata dengan aturan yang telah didefinisikan. Keberadaan karakter berulang yang sering muncul pada data *twitter* dapat menyebabkan metode *stemmer* gagal mengenali kata dasar. Proses normalisasi karakter berulang dicontohkan pada Tabel 3.9 berikut.

Tabel 3.9 Normalisasi Karakter Berulang

Nomor	**Masukan**	**Hasil**
1	minggggguu	minggu
2	berdiriiii	berdiri
3	semangaaat	semangat

Pada penelitian ini, karakter *gg* akan dibiarkan, mengingat banyaknya kata dengan pola tersebut.

3.5 *Stemming* Dengan Metode-Metode *Stemmer* Bahasa Indonesia

Tahap *stemming* adalah proses penguraian kata menjadi bentuk kata dasar dengan menghapus atau mengganti berbagai imbuhan seperti awalan (*prefixes*) dan akhiran (*suffixes*). Metode *stemming* yang digunakan adalah *Porter confix striping stemmer, Nazief Stemmer, Arifin Stemmer, Fadillah Stemmer, Asian Stemmer, Enhanced Confix Stripping Stemmer,* dan *Arifiyanti Stemmer.*

Metode-metode yang digunakan adalah metode *stemmer* berbasis aturan. Metode *porter* merupakan metode yang dikembangkan untuk bahasa Inggris, sedangkan metode lainnya dikembangkan berdasarkan metode *porter* untuk bahasa Indonesia.

3.5.1 Porter Confix Stripping *Stemmer*

Pada metode *porter,* terdapat lima tahap aturan yang dijalankan untuk mendapatkan kata dasar. Setiap aturan dijalankan berurutan, apabila aturan pertama tidak memenuhi kriteria, maka aturan tersebut dilewati dan aturan selanjutnya yang dijalankan. Terdapat empat simbol yang menunjukkan kondisi *stem* yaitu :

1) S jika stem memiliki akhiran S
2) *v* jika stem mengandung huruf *vowel*
3) *d jika stem diakhiri dobel konsonan
4) *o jika stem diakhiri pola konsonan-vokal-konsonan (cvc) selain huruf "X","W", dan "Y".

Aturan yang dijalankan pada metode *porter* yaitu :

1) tahap pertama lakukan penggantian bentuk *past participles* dan *plurals*

   a. sses → ss, ies → i, s → empty

   b. eed → ee, ed → empty, ing → empty

   at → ate, bl → ble, iz → ize

2) Ubah bentuk :

   ational →ate, tional → tion, enci → ence, anci → ance, izer → ize, abli → able, alli→ al, entli → ent, eli → e, ousli → ous, ization → ize, ation → ate , ator → ate, alism → al, iveness → ive, fulness → ful, ousness → ous, aliti → al, iviti → ive, biliti → ble.

3) Ubah bentuk :

   icate → ic, active → *empty,* alize → al, iciti → ic, ical → ic, ful → empty, ness → empty

4) Ubah bentuk :

   al → empty, ance → empty, ence → empty, er → empty, ic → empty, able → empty, ible → empty, ant → empty, ement → empty, ment → empty, ent → empty, ion → empty, ou → empty, ism → empty, ate → empty, iti → empty, ous → empty, ive → empty, ize → empty

Terdapat perhitungan untuk *measure condition* (*m*), yang digunakan sebagai indikator suatu aturan *stemmer* dijalankan atau tidak. Untuk mendapatkan nilai *m* dari suatu kata, dilakukan perhitungan jumlah pasangan huruf vokal dan huruf konsonan. Contohnya : *ization=3,*

3.5.2 Nazief *Stemmer*

Metode *nazief stemmer* dikembangkan berdasarkan metode *porter* untuk Bahasa Indonesia. Metode ini dilakukan dengan menerapkan beberapa aturan pada kata yang akan dicari kata dasarnya, aturan-aturan yang digunakan adalah :

1) Cek kamus kata dasar, bila kata ada pada kamus, proses berhenti.
2) Hapus *Inflectional suffix* yaitu particle suffixes {lah, kah, tah, pun}, dan *possesive pronoun suffixes* {ku, mu, nya}. Lakukan pengecekan kamus kata dasar, jika ditemukan, proses berhenti. Jika tidak ditemukan *Inflectional suffix* atau *pronoun suffixes* lanjutkan tahap berikutnya. Lakukan pengecekan kamus kata dasar.
3) Hapus *Derrivational suffixes* {i, kan, an}. Lakukan pengecekan kamus kata dasar, jika ditemukan, proses berhenti. Jika tidak ditemukan *Derrivational suffixes* lanjutkan tahap berikutnya.
4) Hapus *Derrivational preffixes*{be, di, ke, me, pe, se, te}.
   - Hentikan proses jika :

     1. preffixes yang teridentifikasi membentuk kombinasi yang dilarang.

     2. preffixes yang teridentifikasi sama dengan *preffixes* yang sudah dihapus sebelumnya.

     3. tiga *preffixes* telah dihapus.

- Identifikasi tipe *preffixes*

  1. preffixes {di, ke, se} dapat dihapus langsung.

  2. preffixes {be, te, me, pe} harus dilakukan pengecekan aturan pada tabel 2.1.

5) Apabila semua proses gagal menemukan kata dasar, maka kata yang sudah terhapus karakternya dikembalikan ke bentuk asal. Kata tersebut akan dianggap sebagai kata dasar.

3.5.3 Arifin *Stemmer*

Metode *stemmer arifin* berbeda dengan metode *nazief stemmer,* karena pada metode ini setiap kata diasumsikan memiliki bentuk dua awalan (AW1, AW2) kemudian kata dasar (KD) dan diikuti tiga akhiran (AK1, AK2, AK3). Apabila kata tidak memiliki awalan atau akhiran sebanyak bentuk tersebut, maka awalan yang kosong akan dilabeli dengan "x" sedangkan akhiran yang kosong dilabeli dengan "xx". Tahap-tahap yang dilakukan pada metode *arifin stemmer* yaitu :

1) cek kamus kata dasar, bila kata ada pada kamus, proses berhenti. Jika tidak, lakukan pemotongan AW1.
2) cek kamus kata dasar, bila kata ada pada kamus, proses berhenti. Jika tidak, lakukan pemotongan AW2.
3) cek kamus kata dasar, bila kata ada pada kamus, proses berhenti. Jika tidak, lakukan pemotongan AK1.
4) cek kamus kata dasar, bila kata ada pada kamus, proses berhenti. Jika tidak, lakukan pemotongan AK2.
5) cek kamus kata dasar, bila kata ada pada kamus, proses berhenti. Jika tidak, lakukan pemotongan AK3. Apabila awalan, atau akhiran ada yang kosong, maka kata tidak dirubah.
6) Apabila seluruh proses gagal menemukan kata dasar pada kamus, maka kata yang telah terhapus imbuhannya dikembalikan ke bentuk asal dengan mengkombinasikan kata tersebut dengan awalan atau akhiran yang sudah terhapus.

3.5.4 Fadillah *Stemmer*

Metode *stemmer fadillah* tidak melakukan pengecekan kamus kata dasar, sebagai gantinya digunakan suku kata untuk mengecek apakah kata telah berupa kata dasar. Tahap pertama yang dilakukan adalah memenggal kata berdasarkan suku kata dengan cara :

1) Lakukan pelabelan karakter pada kata dengan simbol "v" untuk vokal, "k" untuk konsonan.
2) Lakukan pengecekan aturan pemenggalan pada kata yang telah dilabeli :
   a. Apabila di tengah kata terdapat dua vokal berturutan (selain diftong), pemisahan dilakukan di antar kedua vokal tersebut.
   b. Apabila di tengah kata terdapat konsonan di antara dua vokal, pemisahan dilakukan sebelum konsonan tersebut
   c. Apabila di tengah kata terdapat dua konsonan atau lebih, pemisahan dilakukan setelah konsonan pertama
   d. Imbuhan dan partikel yang biasanya ditulis serangkai dengan kata dasar, pada penyukuan dipisahkan

Setelah pemenggalan suku kata selesai, dilakukan penghapusan imbuhan dengan tahap berikut :

1) cek jika jumlah suku kata >2 hapus particle ().
2) cek jika jumlah suku kata >2 hapus possesive pronoun ().
3) cek jika jumlah suku kata >2 hapus first order prefix ().
4) cek jika jumlah suku kata >2 dan aturan 3 terpenuhi :
   a. hapus suffix (). jika gagal proses berhenti.
   b. hapus second order prefix () kemudian proses berhenti.
5) cek jika jumlah suku kata >2 dan aturan 3 tidak terpenuhi :
   a. hapus second order prefix ()
   b. hapus suffix () kemudian proses berhenti.

Pada metode ini, setiap aturan dijalankan berurutan. Apabila aturan pertama gagal, aturan selanjutnya akan dijalankan. Apabila pada pengecekan awal sukukata sudah sama dengan atau kurang dari dua, proses *stemmer* tidak dijalankan. Jika setelah penghapusan imbuhan, suku kata sama dengan dua, proses dihentikan.

3.5.5 Asian *Stemmer*

Pada metode *asian stemmer,* langkah-langkah yang dilakukan seperti *nazief stemmer,* tetapi dengan perubahan dan penambahan aturan yaitu :

1) Jika ada bentuk jamak yang sama (buku-buku), maka stem langsung dilakukan pada satu kata saja yaitu "buku". Jika kedua kata setelah tanda sambung "-" berbeda, maka *stemming* dilakukan pada kedua kata tersebut.
2) modifikasi awalan *ter, mem,* dan *meng*.
3) apabila terdapat kombinasi awalan-akhiran *ber-lah, ber-an, mem-i, di-i, pe-i,* dan *ter-i*, maka coba lakukan penghapusan akhiran terlebih dahulu.

3.5.6 Enhanched Confix Stripping *Stemmer*

Pada metode *Enhanched Confix Stripping Stemmer,* langkah-langkah yang dilakukan seperti *asian stemmer,* tetapi dengan perubahan dan penambahan aturan yaitu :

1) penambahan aturan untuk: *mem+p…, men+s…, menge+…, penge+…, peng+k…*

2) penambahan tahap pengembalian akhiran (*loop PengembalianAkhiran*):

   Tahap pengembalian akhiran dimulai dengan mengembalikan kata pada kondisi sebelum *recoding* yaitu sebelum awalan terhapus. Kemudian kembalikan akhiran/sufik yang dimulai dari *derivational suffixes, possesive pronoun,* dan terakhir *inflectional particle.* Pada setiap proses pengembalian, coba lakukan pengecekan kata dasar dengan melakukan penghapusan awalan dengan *recoding*-nya. Jika ditemukan pada kata dasar, maka tahap pengembalian akhiran dihentikan.Apabila sampai pengembalian *inflectional particle* masih belum ditemukan pada kamus kata dasar, maka masukkan awal dianggap kata dasar, dan seluruh tahap dihentikan.

3.5.7*Arifiyanti Stemmer*

Metode *stemmer* ini dikembangkan dari *asian stemmer,* sehingga langkah-langkah yang dilakukan sama dengan *asian stemmer* tetapi dengan penambahan beberapa aturan yaitu :

1) penambahan *derivation suffix*: *-in*
2) penambahan *derivation suffix*:*-ny, -kh, -pn*
3) penambahan *prefix* :*d-*
4) penambahan aturan pada penghapusan *prefix:*

   a. jika huruf awal dari kata yang sudah dihapus awalannya adalah *r-* maka huruf tersebut ikut dihapus.

   b. jika huruf akhir dari kata yang sudah dihapus akhirannya adalah *-k* maka huruf tersebut ikut dihapus.

3.6 *Clustering* Dengan Menggunakan *R*

Setelah selesai melakukan tahap *tokenizing, case folding, stopwords removal, penghapusan karakter berulang,* dan *stemming,* dilakukan tahap pengelompokan data menggunakan metode *k-means* pada *R studio.* Untuk melakukan *clustering* dan evaluasi performa *clustering,* data diubah menjadi format *csv* dan dilakukan penambahan nama variabel serta nomor data. Fungsi untuk menjalankan metode *k-means* telah tersedia pada *R,* namun untuk perhitungan DBI dibutuhkan *library* tambahan yang bernama *clusterSim.*

3.7 Perbandingan Metode Stemmer Dengan Metode Weighted Scoring

Setelah mendapatkan hasil pengujian, dilakukan pemilihan metode *stemmer* terbaik dengan menggunakan metode *weighted scoring.* Untuk itu, perlu dilakukan tahap normalisasi nilai, karena nilai DBI, *understemming, overstemming,* dan waktu proses memiliki skala yang berbeda. Metode normalisasi yang digunakan adalah metode pada sub-bab 2.13.

Pada Tabel 3.10 berikut dijelaskan contoh penerapan metode *weighted scoring* untuk pemilihan metode *stemmer.*

Tabel 3.10 Pemilihan Metode *Stemmer* Dengan Metode Weighted Scoring

Nama Metode	**Kriteria**	**Bobot**	**Skor**	**Nilai**
	Rata-rata DBI	0,45		
	Understemming	0,20		
	Overstemming	0,25		
	Waktu Proses	0,10		
Total Nilai :				

BAB 4
HASIL PENGUJIAN DAN ANALISIS DATA

Pada bab keempat ini akan dijelaskan tentang hasil pengujian yang telah dilakukan dengan menggunakan metode-metode *stemmer* bahasa Indonesia. Proses pengujian yang dilakukan diantaranya adalah pengujian performa metode stemmer dengan menghitung indeks *understemming* dan indeks *overstemming*, pengujian performa *clustering*, serta pengujian signifikansi metode stemmer dengan uji statistik-*t*.

4.1 Uji Data

Data yang diperlukan dalam penelitian ini adalah data komentar pelanggan PLN yang didapatkan dari akun resmi PLN yaitu *@pln123*. Data yang dikumpulkan dimulai dari bulan Agustus sampai September 2016. Metode pengumpulan data yang digunakan adalah dengan menggunakan bantuan twitter API dan perangkat lunak pembantu *R*.

Untuk dapat menggunakan API twitter dibutuhkan kode *api_key* dan *access_token.* Kedua kode tersebut didapat setelah mendaftarkan diri pada situs Twitter Apps yaitu *https://apps.twitter.com/.* Setelah membuat aplikasi baru maka didapatkan empat kode yaitu *api_key, access_token, api_key_secret* dan *access_token_secret.* Contoh dari keempat kode tersebut sebagai berikut:

api_key = "eHVteNUZamVj7nX9Mwz9SDO8H"
api_secret = "XWdlvfKiwdQ1GXh5Ii3fmsD9L27PVrDVjL8HUg6WyuvQ9Pw5Yf"
access_token = "116334468-ayqzHqPti5m6EG14YkmCe05wkM2Bd28I9nYStflN"
access_token_secret= "HYqhoJBbQAKQxrsS7LGX8419Xj413MBUZF5x32uhbQJeH"

Keempat kode tersebut digunakan untuk mengambil data dari *twitter* dengan menggunakan perangkat lunak pembantu yaitu *R*. Sebelum dapat memulai mengumpulkan data dari *twitter,* diperlukan *library StreamR* agar *R* dapat terhubung dengan *twitter.Library* tersebut dapat diunduh dari *https://cran.r-project.org/web/packages/streamR/index.html* atau dapat diimplementasikan langsung pada *R* dengan memberikan kode : *>install.packages(' 'StreamR ',repos = "https://cran.r-project.org").*

Untuk mendapatkan data dari *twitter* PLN, digunakan kata kunci *pln_123,* kemudian data tersebut disimpan dalam format *csv*. Kode lebih detail untuk mengambil data *twitter* pada *R* akan diberikan pada bagian lampiran.

Ada 17 variabel dari data yang tersimpan yaitu *nomor, text, favorited, favoriteCount, replyToSN, created, truncated, replyToSID, id, replyToUID, statusSource, screenName, retweetCount, isRetweet, retweeted, longitude, latitude.* Dari variabel-variabel tersebut,variabel *text* berisi *tweet* pengguna, dan variabel *isRetweet* berisi informasi apakah pesan tersebut di-*retweet.* Contoh dari data yang terkumpul sebagai berikut:

Tabel 4.1 Contoh Data Hasil *CrawlingTwitter*

nomor	text
1	@pln_123 jangan mencontoh pln di daerah lain min, kita paling bermasalah disini. Insiatif sedikit, buat sesuatu yang bermanfaat.
2	@pln_123 haruskah ane kerja disana supaya masyarakat ga marah2 lg dengan pemadaman listrik seperti minum obat tanpa adanya pemberitahuan.
3	@pln_123 Harusnya pln buat aplikasi notifikasi dini pemadaman listrik. Setiap mau pemadaman masyarakat di berikan pln berupa sms gateway.
4	@pln_123 merubah nama ke pemilikan misalkan ini meteran punya atas nama bpk saya...trus sya ganti jadi nama saya gitu pak?
5	RT @mazwitazza: @pln_123 begitu ya pk...trus itu pak masalah surat kuasa untuk tambah daya bisa di dapat di mana bpk?

Data yang telah tersimpan dalam format *csv* tersebut perlu dilakukan pra-proses data. Langkah pertama adalah mengambil data keluhan dari variabel *text,* sedangkan 16 variabel lainya tidak digunakan. Kemudian dilakukan pembersihan *retweet,* yaitu *tweet* dari pengguna yang diteruskan ke pengguna *twitter* yang lain. Dari proses tersebut didapatkan 2730 data *tweet* yang akan digunakan dalam pengujian. Data tersebut disimpan dalam format *txt* untuk dibaca oleh program yang dibuat.

Pada program yang dibuat dilakukan tahap *tokenizing, case folding, stopwords removal, penghapusan karakter berulang,* dan *stemming.* Data yang telah diproses pada program disimpan dalam format *txt.*Untuk melakukan *clustering* dan evaluasi performa *clustering,* data yang tersimpan dalam format *txt* tersebut diubah kembali menjadi format *csv* dengan penambahan nama variabel dan nomor data.

4.2 Lingkungan Pengujian

Lingkungan untuk pengujian meliputi perangkat lunak dan spesifikasi perangkat keras yang digunakan untuk menguji performa metode-metode stemmer pada penelitian ini. Untuk spesifikasi perangkat keras yang digunakan, dapat dilihat pada Tabel 4.2 berikut

Tabel 4.2 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi
Jenis	Notebook
Prosesor	Intel® Core™ i3 M350 @2.27 GHz
RAM	4 GB

Sedangkan untuk perangkat lunak yang digunakan untuk penelitian ini dapat dilihat pada Tabel 4.3 berikut.

Tabel 4.3 Spesifikasi Perangkat Lunak

Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows 7 Home Premium 64-bit
Bahasa Pemrograman	• Java • SQL • R
Tools	• XAMPP Version : 5.6.12 • Java Version 8 : 1.8.0_60 • NetBeans IDE : 7.3.1 • R StudioVersion : 1.0.136

4.3 Hasil Skenario Pengujian

Skenario pengujian dilakukan untuk mengetahui performa metode stemmer bahasa Indonesia dan dampaknya pada hasil pengelompokan data teks (*clustering*). Uji coba yang dilakukan dalam penelitian ini adalah perbandingan performa metode stemmer, perbandingan performa *clustering* setiap metode stemmer, dan pengujian signifikansi dengan *paired t test.* Untuk masing-masing pengujian akan dijelaskan lebih lanjut pada sub-bab berikutnya.

4.3.1 Hasil Pengujian Performa Metode Stemmer

Pengujian ini dilakukan untuk mengetahui bagaimana performa setiap metode stemmer dalam mengubah kata menjadi kata dasar dengan menghitung jumlah kesalahan kata dasar yang terjadi. Kesalahan dalam *stemmer* dapat diukur dengan menghitung nilai indeks *overstemming*, dan indeks *understemming*. *Overstemming* terjadi apabila metode *stemmer* salah dalam menghapus imbuhan dari kata dasar, dimana kata dasar ikut terhapus sebagian dan menyebabkan dua kata atau lebih yang tidak berkaitan menjadi satu kata dasar. Sedangkan *understemming* terjadi apabila metode *stemmer* tidak sempurna dalam menghapus imbuhan dimana sebagian imbuhan masih bergabung dengan kata dasarnya.

Untuk dapat melakukan perhitungan *overstemming* dan *understemming,* maka dibutuhkan data yang berisi daftar kata-kata yang memiliki kata dasar yang sama. Kata-kata tersebut dikelompokkan menjadi kelompok kata.Cara mendapatkan kelompok-kelompok kata tersebut dilakukan dengan dua tahap yaitu tahap pengambilan kata dari hasil pra-proses data dan tahap pembuatan kelompok kata. Tahap pengambilan kata dilakukan dengan membuat kode program yang berfungsi mengecek apakah kata adalah kata berimbuhan. Setelah kata-kata berimbuhan dikumpulkan, dilakukan pengambilan dan pembuatan kelompok kata secara manual. Kedua tahap tersebut dilakukan karena belum tersedianya daftar kelompok kata untuk Bahasa Indonesia yang siap digunakan untuk pengujian *overstemming* dan *understemming*. Pada penelitian ini, data yang digunakan untuk pengujian performa metode *stemmer* berjumlah 936 kata yang terbagi menjadi 200 kelompok kata. Sedangkan jumlah kata setiap kelompok berada pada kisaran 2 kata sampai dengan 8 kata. Contoh data kelompok kata ada pada Tabel 4.4 sedangkan contoh data kelompok kata yang telah di-*stemming* disajikan pada Tabel 4.5.

Tabel 4.4 Contoh Data Kelompok Kata

Kelompok	Kata
1	akhir akhirnya pengakhiran diakhiri
2	balas dibalas pembalasannya terbalaskan membalaskan
3	cepat secepatnya kecepatan dipercepat
4	dapat mendapat mendapati pendapatannya didapatinya
5	ganti mengganti pergantian menggantikan tergantikan

Tabel 4.5 Contoh Data Kelompok Kata Ter-*stemming*

Kelompok	Kata
1	akhir akhirakhirakhir
2	balas balasbalasbalasbalas
3	cepat secepatnya cepatcepat
4	dapat dapat mendapati dapat didapatinya
5	ganti gantigantigantiganti

Kode program tahap pengambilan kata dan contoh keluaran program disertakan pada bagian lampiran.

Untuk mendapatkan nilai *overstemming, understemming,* dan *stemming weight* maka dibuat kode program dengan menerapkan langkah-langkah dan persamaan yang telah dijabarkan pada sub-Bab II tentang evaluasi metode *stemmer.* Langkah-langkah yang dilakukan meliputi perhitungan nilai *desired merge total* (DMT), *desired non-merge total* (DNT), *global wrongly-merged total* (GWMT) dan *global desired non-merge total* (GDNT). Kemudian dilakukan perhitungan untuk mendapatkan nilai *wrongly-merged total* (WMT), *global wrongly-merged total* (GWMT), *unachieved-merge total* (UMT), dan *global unachieved-merge total* (GUMT). Setelah proses perhitungan tersebut, nilai *overstemming, understemming,* dan *stemming weight* bisa didapatkan. Tabel 4.6 berikut adalah nilai indeks *overstemming* dan *understemming* untuk setiap metode *stemmer.*

Tabel 4.6 Indeks *Overstemming, Understemming,* Dan *Stemming Weight*

Nomor	Metode Stemmer	***Overstemming Index***	***Understemming Index***	***Stemming Weight***
1	*Porter confix stripping stemmer*	0,0	0,226615	0,0
2	*Nazief Stemmer*	0,001124	0,104166	0,010796
3	*Arifin Stemmer*	0,001103	0,110969	0,009948
4	*Fadillah Stemmer*	0,001110	0,132228	0,008400
5	*Asian Stemmer*	0,001152	0,087585	0,013155
6	*Enhanced Confix Stripping Stemmer*	0,001166	0,068452	0,017034
7	*Arifiyanti Stemmer*	0,001166	0,086309	0,013509

Kemudian sebagai tambahan, pada Tabel 4.7 berikut ini akan ditunjukkan waktu yang dibutuhkan masing-masing metode *stemmer* dalam memproses data

yang digunakan. Untuk mengukur waktu proses metode stemmer dilakukan menggunakan kode program *System.nanotime()* yang akan memberikan nilai dengan satuan nano detik. Untuk mendapatkan waktu yang dibutuhkan oleh metode *stemmer* digunakan rumus :

waktu proses = (waktu proses selesai – waktu proses dimulai) : 10^9

dimana :
waktu proses : waktu yang dibutuhkan metode *stemmer* untuk memproses data.
waktu proses selesai : waktu saat data selesai di-*stemming*.
waktu proses dimulai : waktu saat data mulai di-*stemming*.
10^9 : bilangan untuk mengkonversi nano detik menjadi detik.

Tabel 4.7 Waktu Pengujian Metode-Metode *Stemmer*

Nama Metode	Waktu Proses (Detik)	Waktu Proses *Paice Evaluation* (Detik)
Porter confix stripping stemmer	0,627	0,033
Nazief Stemmer	1961,879	119,430
Arifin Stemmer	3624,804	181,423
Fadillah Stemmer	0,218	0,029
Asian Stemmer	1967,301	119,415
Enhanced Confix Stripping Stemmer	2702,450	142,134
Arifiyanti Stemmer	2416,499	123,805

4.3.2 Hasil Pengujian Performa Pengelompokan Data

Pengujian ini dilakukan untuk mengetahui metode *stemmer* mana yang paling baik diterapkan dengan melihat dampak metode stemmer tersebut pada performa pengelompokan data keluhan pelanggan yang digunakan. Untuk metode pengelompokan data yang digunakan adalah *k-means* yang merupakan salah satu metode *clustering* yang digunakan secara luas di berbagai bidang. Sedangkan untuk mengukur performa metode *clustering* digunakan *davies-bouldin index* (DBI). Nilai DBI yang lebih rendah adalah solusi *clustering* yang lebih baik, sehingga dapat disimpulkan metode *stemmer* bahasa Indonesia apa yang paling baik diterapkan.

Pada pengujian ini akan dibandingkan bagaimana dampak penerapan metode *stemmer* terhadap performa hasil *clustering*. Metode-metode stemmer

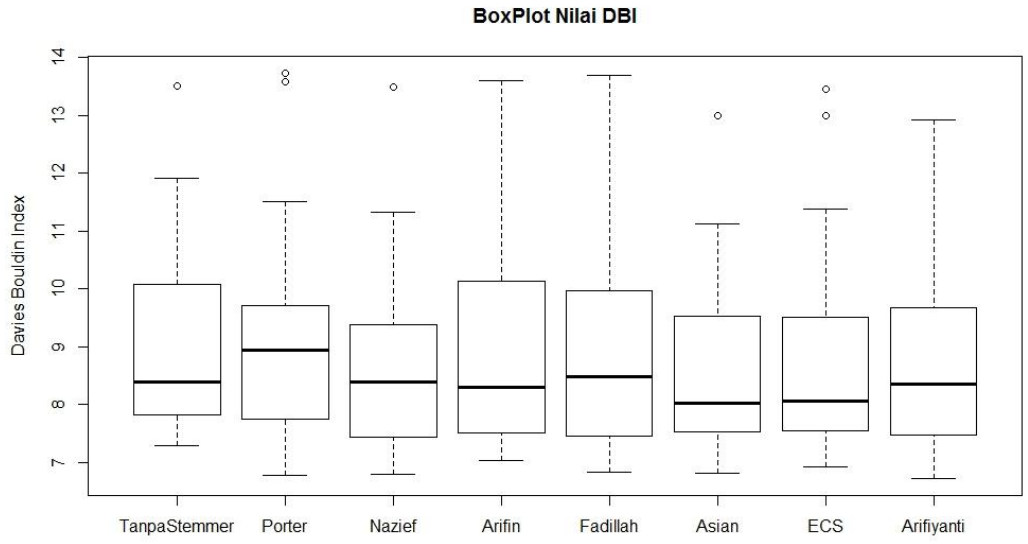
yang diuji adalah semua metode yang telah dijelaskan pada Bab 1 bagian pendahuluan. Untuk metode pengelompokan data digunakan *k-means* dan metode evaluasi menggunakan DBI. Untuk melakukan pegelompokan data dan menghitung nilai DBI, digunakan perangkat lunak pembantu yaitu *R*. Fungsi untuk menjalankan metode *k-means* telah tersedia pada *R*, namun untuk perhitungan DBI dibutuhkan *library* tambahan yang bernama *clusterSim*. Library tersebut dapat diunduh pada situs :
https://cran.r-project.org/web/packages/clusterSim/index.html.

Langkah teknis yang lebih detail untuk melakukan pengelompokan data dan menghitung nilai DBI berserta kode program *R* akan diberikan pada bagian lampiran. Pada Tabel 4.8 berikut akan ditunjukkan hasil pengujian performa pengelompokan data.

Tabel 4.8 Hasil Perhitungan DBI Pengelompokan Data

Jum-lah Kelas	**Tanpa Stemm-er**	***Arifiyan-ti***	***Enhanced Confix Stripping***	***Asian***	***Fadillah***	***Arifin***	***Nazief***	***Porter confix stripping***
2	13,510	**12,926**	13,458	13,001	13,717	13,605	13,482	13,572
3	11,905	10,601	12,983	11,129	11,140	**10,475**	11,321	13,734
4	10,387	11,696	11,375	**10,253**	11,584	10,633	10,968	11,510
5	10,441	10,159	**9,017**	10,568	11,497	10,630	10,218	10,566
6	9,338	9,615	9,885	9,451	9,785	10,013	**9,330**	9,897
7	10,030	9,733	9,412	9,595	9,870	10,252	**9,406**	9,412
8	10,138	**8,878**	9,598	9,418	9,320	9,257	8,960	9,540
9	9,430	8,361	8,253	8,832	9,290	**8,025**	8,394	9,255
10	8,841	8,561	**7,743**	8,537	8,891	9,110	8,676	9,459
11	8,388	7,693	8,232	**7,673**	8,486	8,442	8,787	8,949
12	8,131	8,366	**7,623**	8,025	8,076	7,810	7,715	7,699
13	7,925	7,635	7,544	7,782	**7,288**	8,295	7,970	8,296
14	7,967	7,526	8,067	**7,162**	7,657	7,702	7,431	7,986
15	8,073	7,749	7,306	7,555	**7,274**	7,455	7,434	8,090
16	7,721	**6,885**	7,371	7,830	7,423	7,579	7,954	7,528
17	7,471	7,397	7,554	7,514	**7,141**	7,436	7,295	7,810
18	7,478	7,438	7,413	**6,926**	7,200	7,027	7,034	7,656
19	7,309	6,845	7,543	6,808	6,881	7,234	**6,795**	7,334
20	7,295	**6,717**	6,922	7,163	7,077	7,083	7,243	6,772
Rata-rata	9,041	**8,673**	8,805	8,696	8,926	8,845	8,759	9,214

Berdasarkan nilai-nilai DBI pada Tabel 4.8, dibuat *boxplot* yang memberikan gambaran rentang nilai DBI setiap metode *stemmer*. Gambar 4.1 berikut menunjukkan *boxplot* yang terbentuk. Perbandingan gambar *boxplot* yang lebih detail disertakan pada bagian lampiran, termasuk *stemmer* modifikasi.



Gambar 4.1 *Boxplot* dari nilai DBI metode *stemmer*

4.3.3 Hasil Pengujian Signifikansi Metode Stemmer

Pengujian ini dilakukan untuk mengetahui apakah performa pengelompokan data yang didapat dengan satu metode *stemmer* berbeda signifikan dengan hasil yang didapat metode *stemmer* yang lain. Untuk mengetahui signifikansi tersebut digunakan uji statistik *paired t test*. Masukan untuk melakukan uji *t* tersebut adalah nilai DBI yang didapatkan pada saat pengujian performa pengelompokan data. Selain perbandingan signifikansi antar metode *stemmer*, juga akan dilihat signifikansi penerapan metode *stemmer* dengan proses yang tanpa menggunakan metode *stemmer*.

Pengujian signifikansi untuk metode-metode *stemmer* dilakukan untuk mengetahui apakah peningkatan performa hasil *clustering* yang didapat signifikan. Pengujian dilakukan dengan perhitungan nilai *p-value* dengan *convidence level* 95%. Nilai *p-value* dibawah 0,05 menunjukkan adanya perbedaan signifikan antara kedua kelompok data. Pada Tabel 4.9 berikut, pembacaan kesimpulan

dimulai dari baris menuju kolom. Signifikansi ditandai dengan simbol "+" sedangkan simbol "-" menyatakan sebaliknya yaitu metode pada baris memiliki performa lebih buruk secara signifikan dari metode yang ada pada kolom tabel. Apabila tidak ada signifikansi maka digunakan simbol "=". Contoh cara pengisian tabel pengujian signifikansi sebagai berikut :

apabila nilai *p-value* lebih dari 0,05 maka diberikan simbol "=", misal antara *tanpa stemmer* dengan *nazief stemmer* didapatkan hasil *p-value* = 0,01405 dan *t* = 2,7197. Karena nilai *p-value* kurang dari 0,05 dan *t* bernilai positif, maka disimpulkan perbedaan antara *azief stemmer* dan *tanpa stemmer* adalah "berbeda signifikan, positif" sehingga pada kolom diberikan simbol "-". Artinya, *nazief stemmer* lebih baik dari *tanpa stemmer*. Sementara untuk *tanpa stemmer* dengan *fadillah stemmer* didapatkan *t* = 0,62829 dan *p-value* = 0,5377. Karena nilai *p-value*lebih dari 0,05 dan *t* bernilai positif, maka disimpulkan perbedaan antara *tanpa stemmer* dengan *fadillah stemmer* adalah "tidak berbeda, positif" sehingga diberikan simbol "=". Artinya walaupun *fadillah stemmer* memiliki hasil lebih baik dari *tanpa stemmer*, namun perbedaan yang ada tidak signifikan. Untuk *enhanced confix stripping* dan *porter confix stripping* didapatkan *t* = -3,1676, dan *p-value* = 0,005329. Karena nilai *p-value* kurang dari 0,05 dan *t* bernilai negatif, maka disimpulkan perbedaan antara *enhanced confix stripping* dan *porter confix stripping* adalah "berbeda signifikan, negatif" sehingga pada kolom diberikan simbol "+". Artinya *porter confix stripping* memiliki perbedaan signifikan yang lebih buruk dibandingkan *enhanced confix stripping*.

Tabel 4.9 Hasil Pengujian Signifikansi Metode Stemmer

	Tanpa Stemmer	***Arifiyanti***	***Enhanced Confix Stripping***	***Asian***	***Fadillah***	***Arifin***	***Nazief***	***Porter confix stripping***
Tanpa Stemmer		-	=	-	=	=	-	=
Arifiyanti			=	=	=	=	=	+
Enhanced Confix Stripping				=	=	=	=	+
Asian					=	=	=	+
Fadillah						=	=	+
Arifin							=	=
Nazief								+

4.4 Analisis Hasil Pengujian

Setelah melakukan skenario pengujian, didapatkan berbagai nilai dan hasil. Pada sub-bab analisis hasil pengujian ini akan dibahas mengenai nilai dan hasil yang telah didapatkan.

4.4.1 Analisis Hasil Pengujian Performa Metode Stemmer

Pada bagian ini akan dijabarkan analisis dari hasil yang didapatkan pada Tabel 4.6 tentang indeks *overstemming, understemming,* dan *stemming weight* dan pada Tabel 4.7 tentang waktu pengujian metode-metode *stemmer.*

Dimulai dengan *porter confix stripping stemmer,* metode tersebut mendapatkan nilai *understemming* paling besar yaitu 0,225. Artinya metode tersebut tidak mempu men-*stemming* kelompok kata berimbuhan yang memiliki kata dasar sama, menjadi satu kata. Contohnya : kelompok *"akhir akhirnya pengakhiran diakhiri"* menjadi *"akhir akhir akhir akhir".* Hal ini disebabkan karena aturan *porter confix stripping stemmer* yang tidak cocok dengan prefiks maupun sufiks kata berimbuhan yang ada dan menyebabkan kata berimbuhan tidak ter-*stemming.* Metode *porter confix stripping stemmer* memiliki aturan untuk penghapusan suffiks *s* misalnya kata *cats* menjadi cat, namun aturan ini berdampak pada kelompok kata *"protes diprotes"*. Kelompok tersebut seharusnya ter-*stemming* menjadi *"protes protes"* tetapi dengan aturan penghapusan *s*, kata *protes* menjadi *prote*. Sedangkankata *diprotes* menjadi *diprote* kemudian menjadi *diprot* karena adanya aturan penghapusan *e* ditahap selanjutnya. Contoh lain yang ditemukan dari data hasil pengujian adalah kata *jaring* dan *terjaring.* Kata *jaring* menjadi *jare* karena aturan penghapusan suffiks *ing* kemudian aturan penambahan *e* yang sebenarnya diperuntukkan untuk kata seperti *fil(ing)* menjadi *file.*

Kesalahan yang terjadi tersebut membuat kelompok kata yang sebenarnya memiliki kata dasar sama tidak ter-*stemming* menjadi satu kata. Ditambah lagi aturan-aturan yang tidak cocok untuk kata seperti *"menghidupkan penghidupan dihidupkan"*. Hal tersebut menyebabkan nilai *understemming*untuk metode *porter confix stripping stemmer* menjadi yang terbesar. Sebaliknya, karena tidak ada kata-kata yang memiliki kata dasar berbeda tetapi ter-*stemming* menjadi satu kata (*kehidupan* dan *hidung* menjadi *hidu),* maka nilai *overstemming* menjadi 0.

Karena metode *porter confix stripping stemmer* hanya menggunakan aturan-aturan dan pengecekan *meassure,* maka waktu yang dibutuhkan untuk memproses data sangat cepat. Waktu yang dibutuhkan untuk memproses seluruh data *twitter* adalah 0,627 detik sedangkan untuk memproses data kelompok kata yang disiapkan hanya 0,033 detik. Waktu tersebut adalah yang tercepat kedua setelah metode *fadillah stemmer*.

Metode kedua yang diuji adalah *nazief stemmer*. Metode tersebut dikembangkan berdasarkan *porter confix stripping stemmer* yang aturan-aturannya dirancang untuk bahasa Indonesia. Perbedaan diantara kedua metode tersebut selain aturan-aturan yang digunakan, juga terdapat perbedaan pada pengecekan *meassure* yang berfungsi sebagai kontrol agar tidak memotong kata yang sudah sangat pendek misalnya *owl, tree,* dan *roll.* Pada metode *porter confix stripping stemmer, meassure* dilakukan dengan pengecekan jumlah pasangan vokal-konsonan. Sementara pada metode *nazief stemmer*, kontrol dilakukan dengan pengecekkan kamus kata dasar.

Metode *nazief stemmer* memiliki nilai *understemming* 0,104. Nilai tersebut lebih baik dari metode *porter confix stripping stemmer*. Artinya metode *nazief stemmer* mampu menyatukan kata-kata pada suatu kelompok kata dasar dengan baik. Contoh berikut didapatkan dari hasil pengujian metode *nazief stemmer* :
*"adu pengaduan diadukan peraduan"*menjadi *"adu adu adu adu"*.

Namun demikian, ditemukan kesalahan *stemming* pada kelompok *"arti berarti mengartikan artinya pengartian diartikan"*menjadi *"arti berarti arti arti arti arti"*. Dapat dilihat bahwa metode *nazief stemmer* tidak men-*stemming* kata *berarti*. Hal tersebut disebabkan oleh urutan penghapusan imbuhan, dimana pada pada proses *hapus derivational suffix* kata *berarti* diubah menjadi *berart.* Karena tidak ditemukan pada kamus, kata *berart* diubah menjadi *art* pada proses *hapus preffix*. Karena tidak ditemuakan lagi pada kamus, proses *recoding* dilakukan sehingga kata *berart* diubah menjadi *rart.* Seluruh tahap metode *nazief* gagal menemukan kata pada kamus kata dasar sehingga kata dikembalikan menjadi *berarti*.

Contoh kesalahan kedua yang ditemukan pada kelompok kata *"hari sehari seharipun seharian"*menjadi *"hari sehari seharipun hari"*. Dari hasil tersebut

terlihat bahwa kata *sehari* dan *seharipun*tidak ter-*stemming*. Kata *seharipun* melalui proses *hapus inflectional particle* menjadi *sehari,* kemudian karena tidak ditemukan pada kamus kata dasar, proses *hapus derivational suffix* dilakukan untuk mengubah kata *sehari* menjadi *sehar.* Karena masih belum ditemukan pada kamus kata dasar, kata *sehar* melalui proses penghapusan prefik menjadi *har.* Karena prefik *se* tidak termasuk aturan *recoding,* maka seluruh proses gagal mengenali kata dasar dan kata *seharipun* dianggap kata dasar.

Kesalahan tersebut membuat kelompok kata *"hari sehari seharipun seharian"* tidak menjadi satu kata dan menimbulkan kesalahan *understemming.* Jika dilihat perbandingan nilai *understemming* dan *overstemming* metode *nazief stemmer,* yaitu 0,104 dan 0,001, maka dapat disimpulkan bahwa kesalahan metode tersebut mayoritas ada pada *understemming.* Metode *nazief* tidak sampai membuat kesalahan-kesalahan *overstemming.* Rendahnya nilai *overstemming* dapat disebabkan oleh penggunaan kamus kata dasar untuk mengecek setiap hasil penghapusan imbuhan.

Untuk waktu pemrosesan data *twitter,* metode *nazief* membutuhkan waktu 1962 detik, sedangkan waktu pemrosesan kelompok kata berimbuhan 119 detik. Waktu tersebut jauh lebih lama dibandingkan metode *porter confix stripping stemmer* dalam memproses data yang sama. Hal tersebut disebabkan oleh jumlah aturan yang ada lebih banyak dari metode *porter confix stripping,* selain itu juga disebabkan karena penggunaan kamus kata dasar bahasa Indonesia. Kamus kata dasar yang digunakan berjumlah 28.500 kata, sedangkan pada metode *nazief stemmer* terdapat total enam kali proses pengecekan kamus kata dasar.

Metode ketiga yang dibahas adalah *arifin stemmer.* Metode tersebut juga menggunakan kamus kata dasar seperti metode *nazief stemmer.* Pada penelitian ini, semua metode *stemmer* yang menggunakan kamus kata dasar bahasa Indonesia, menggunakan satu kamus yang sama, yaitu berjumlah 28.500 kata.

Metode *arifin stemmer* sebenarnya lebih kompleks dari metode *nazief,* karena adanya proses mengkombinasikan imbuhan yang telah dihapus dengan kata dasar, apabila kata dasar masih belum ditemukan pada kamus kata dasar.

Metode *arifin stemmer* memiliki nilai hasil pengujian *understemming* sebesar 0,111 dan nilai *overstemming* 0,001. Nilai yang didapatkan menunjukkan

bahwa metode *arifin* memiliki performa dibawah metode *nazief stemmer*. Hal ini disebabkan oleh adanya beberapa kesalahan dalam men-*stemming* kata. Contohnya pada kelompok kata *"berangkat keberangkatan diberangkatkan berangkatnya"* menjadi *"berangkat angkat angkat angkat"*. Permasalahan timbul bukan hanya karena aturan *stemming* yang digunakan, namun juga disebabkan oleh kata *berangkat* dan *angkat*. Kedua kata tersebut merupakan kata dasar yang valid. Untuk kata *berangkat*, mula-mula metode *arifin* akan mengecek kamus kata dasar dan menemukan bahwa kata tersebut valid. Namun untuk kata *keberangkatan*, tidak ditemukan dalam kamus, sehingga dilakukan penghapusan prefik pertama menjadi *berangkatan*. Karena belum ditemukan, maka dilakukan penghapusan prefik kedua, yang mengubah kata *berangkatan* menjadi *angkatan*. Langkah selanjutnya adalah penghapusan akhiran pertama, yang mengubah kata *angkatan* menjadi *angkat*. Kata *angkat* ditemukan pada kamus, dan seluruh proses dihentikan. Hal tersebut juga terjadi pada kata *diberangkatkan* dan *berangkatnya*. Kesalahan tersebut menyebabkan metode *arifin stemmer* mengenali kata *berangkat* sebagai katadasar, tetapi tidak untuk kata *keberangkatan*, *diberangkatkan* dan *berangkatnya*. Kesalahan yang terjadi menyebabkan nilai understemming bertambah besar karena kelompok kata berimbuhan tidak ter-*stemming* menjadi satu kata yang sama.

Kesalahan tersebut tidak terjadi pada metode *nazief stemmer*, karena pada metode *nazief* kata *keberangkatan* akan dihapus sufik terlebih dahulu menjadi *keberangkat*. Kemudian dilakukan proses penghapusan prefik yang mengubah kata *keberangkat* menjadi *berangkat*. Karena kata *berangkat* ditemukan pada kamus proses berhenti dan tidak sampai menghapus prefik *ber*. Metode *nazief* mampu mengubah kelompok kata tersebut menjadi satu kata yaitu *"berangkat berangkat berangkat berangkat"*.

Untuk waktu pemrosesan data twitter, metode *arifin stemmer* membutuhkan waktu 3625 detik. Sedangkan waktu pemrosesan kelompok kata membutuhkan 181 detik. Waktu yang dibutuhkan untuk memproses data adalah yang paling lama diantara metode-metode *stemmer* yang diuji. Waktu yang lama tersebut disebabkan oleh jumlah pengecekan kamus kata dasar yang lebih banyak dibandingkan metode *nazief*, yaitu pada proses pengecekan awal, proses

penghapusan prefik dan sufik, proses kombinasi prefik dan sufik terhadap kata dasar, dan proses *recoding*. Total terdapat 13 kali pengecekan kamus kata dasar.

Metode selanjutnya adalah *fadillah stemmer* yang juga dikembangkan dari *porter confix stripping stemmer*. Metode *fadillah* sama dengan metode *porter confix stripping stemmer* yang tidak menggunakan kamus kata dasar. Sebagai gantinya, metode *fadillah* menggunakan suku kata yang fungsinya sama dengan *measure* pada metode *porter confix stripping stemmer*. Dengan adanya suku kata, metode *fadillah* akan lebih berhati-hati dengan tidak menghapus kata yang sudah terlalu pendek, contohnya *beras, teri*, dan *ampun.*

Untuk indeks *understemming,* metode *fadillah stemmer* mendapatkan nilai 0,132dan nilai *overstemming*0,001. Nilai tersebut lebih tinggi dibandingkan metode *arifin stemmer,* tetapi lebih rendah dari metode *porter confix stripping stemmer.* Nilai *understemming* yang lebih rendah tersebut disebabkan diantaranya oleh urutan aturan penghapusan imbuhan dan tidak adanya kamus kata dasar bahasa Indonesia sebagai kontrol. Kesalahan *stemmer* yang terjadi adalah pada kelompok kata *"akui mengakui diakuinya pengakuannya"*. Metode *fadillah* menghasilkan kelompok kata *"aku kaku aku aku",* yang menyebabkan bertambahnya indeks *understemming.* Sedangkan untuk kelompok kata *"adu pengaduan diadukan peraduan",* metode *fadillah* menghasilkan *"adu adu adu radu"*. Kedua kesalahan tersebut tidak terjadi pada metode *arifin* dan *naziefstemmer.*

Jika dianalisis lebih lanjut untuk kata *mengakui*, metode *fadillah* akan langsung menjalankan tiga proses yaitu *hapus partikel(lah, kah, pun), hapus possesive pronoun(ku, mu, nya),* dan *hapus first order prefix(me, pe, ter, ke, di, dst.).* Pada tahap *hapus partikel* dan *hapus possesive pronoun,* kata *mengakui* tidak berubah, namun pada tahap *hapus first order prefix*, kata *mengakui* diubah menjadi *kakui.* Karena terdapat imbuhan yang berhasil dihapus, dan jumlah suku kata dari kata *kakui* lebih dari 2, maka tahap *hapus sufix(kan, i, an)* dilakukan. Tahap *hapus sufix* tersebut mengubah kata *kakui* menjadi *kaku.* Setelah berhasil menjalankan tahap *hapus sufix*, tahap *hapus second order prefix(be, ber, pe, pel, dst.)* dilakukan. Tetapi tahap *hapus second order prefix* tidak berdampak pada kata *kaku*, sehinga kata *kaku* dianggap hasil akhir.

Untuk kata *peraduan,* metode *fadillah* juga menjalankan tahap *hapus partikel, hapus possesive pronoun,* dan *hapus first order prefix* karena jumlah suku kata dari kata *peraduan* lebih dari 2. Tetapi ketiga tahap tersebut gagal menghapus imbuhan, sehingga tahap *hapus second order prefix* dilakukan, diikuti dengan tahap *hapus sufix.* Kedua tahap tersebut akan mengubah kata *peraduan* menjadi *raduan,* dan karena jumlah suku kata dari *raduan* masih lebih dari 2, kata tersebut diubah menjadi *radu.*

Dari penjabaran tersebut, dapat disimpulkan bahwa permasalahan terjadi pada tahap *hapus first order prefix* dan *hapus second order prefix.* Pada kasus *hapus first order prefix,* kata *mengakui* diubah menjadi *kakui.* Sedangkan pada kasus *hapus second order prefix,* kata *peraduan* diubah menjadi *raduan.* Karena metode *fadillah stemmer* tidak menggunakan proses pengecekkan kamus kata dasar, maka proses *recoding* menjadi lebih sulit untuk diterapkan.

Waktu yang dibutuhkan metode *fadillah stemmer* untuk memproses seluruh data *twitter* dan data dari kelompok kata adalah yang paling cepat, yaitu 0,218 detik dan 0,029 detik. Waktu yang dibutuhkan metode *porter confix stripping stemmer* lebih lama karena banyaknya aturan-aturan yang tidak cocok dengan kata berimbuhan bahasa Indonesia. Hal tersebut menyebabkan program melakukan pengecekkan seluruh aturan yang ada pada metode *porter confix stripping stemmer.*

Metode *stemmer* selanjutnya adalah *asian stemmer.* Metode tersebut merupakan hasil penyempurnaan dari metode *nazief stemmer.* Pada metode *asian* terdapat proses pengecekkan *rule precedence,* yang akan mengubah urutan penghapusan prefik dan sufik. Pada metode *nazief stemmer,* seluruh kata akan diproses dengan tahap-tahap penghapusan sufik, kemudian diikuti tahap penghapusan prefik, namun pada metode *asian stemmer* diberikan beberapa kondisi yang apabila kondisi tersebut terpenuhi, maka tahap penghapusan prefik dilakukan terlebih dahulu, kemudian diikuti tahap penghapusan sufik. Kondisi khusus tersebut yang disebut sebagai *rule precedence.*

Pada Tabel 4.5 didapatkan nilai *understemming* dan *overstemming* untuk metode *asian stemmer* adalah 0,087 dan 0,001, sementara nilai *stemming weight*

0,013. Dari hasil tersebut dapat disimpulkan bahwa metode *asian* lebih unggul dibandingkan metode *nazief stemmer*.

Jika diamati pada hasil pengujian, maka didapatkan kesimpulan bahwa jenis kesalahan yang ada pada metode *asian* dan metode *nazief* hampir sama, misalnya pada kata *berarti* dan *seharipun* yang telah dibahas sebelumnya. Metode *asian stemmer* juga tidak dapat menangani kesalahan tersebut. Tetapi ada kesalahan-kesalahan yang terjadi pada metode *nazief,* namun tidak terjadi pada metode *asian.* Contohnya pada kelompok kata "*masalah permasalahan permasalahannya bermasalah dimasalahkan masalahnya"*. Metode *asian* mampu mengubah semua kata tersebut menjadi satu kata yaitu *masalah,* sedangkan metode *nazief* mengubah kata *bermasalah* menjadi *masa.* Contoh lainya pada kelompok kata *"capai mencapai pencapaian tercapai"*, dimana metode *nazief* mengubah kata *mencapai* dan kata *tercapai* menjadi *capa.* Sedangkan metode *asian* mampu mengubah semua kata tersebut menjadi *capai.*

Keberhasilan metode *asian* dalam men-*stemming* kata *bermasalah, mencapai,* dan *tercapai* karena adanya proses pengecekan *rule precedence.* Karena persyaratan *rule precedence* untuk ketiga kata tersebut terpenuhi, maka penghapusan prefik dilakukan terlebih dahulu sebelum melakukan penghapusan sufik. Misalnya, untuk kata *bermasalah,* metode *asian* akan melakukan penghapusan prefik yang mengubah kata *bermasalah* menjadi *masalah.* Karena *masalah* ditemukan pada kata dasar, maka proses dihentikan. Proses yang sama juga terjadi pada kata *mencapai* dan *tercapai.* Namun pada metode *nazief,* kata *bermasalah* diubah menjadi *bermasa,* kemudian dilanjutkan dengan proses penghapusan prefik menjadi *masa.* Kata *masa* terdapat di kata dasar sehingga proses dihentikan. Tetapi kata *masa* bukan kata dasar yang diinginkan.

Kesalahan yang terjadi pada metode *asian* untuk kata *berarti* dan *seharipun* disebabkan oleh tidak adanya aturan *rule precedence* untuk kedua kata tersebut. Misal dilakukan penambahan aturan *ber-i* dan *se-pun*, maka kata *berarti* akan lansung diubah menjadi *arti*. Sedangkan untuk kata *seharipun* akan dilakukan penghapusan prefik menjadi *haripun,* kemudian dilakukan penghapusan sufik menjadi *hari.*

Untuk waktu proses yang dibutuhkan metode *asian stemmer* adalah 1967detik untuk memproses seluruh data *twitter,* dan 119 detik untuk memproses data kelompok kata berimbuhan. Waktu yang dibutuhkan tersebut lebih lama dari waktu yang dibutuhkan metode *nazief stemmer* tetapi lebih cepat daripada metode *arifin stemmer*. Hal tersebut disebabkan adanya penambahan proses untuk pengecekkan *rule precedence* dan penambahan aturan penghapusan prefik.

Metode yang ke enam adalah *enhanced confix stripping stemmer.* Metode tersebut merupakan pengembangan dari metode *asian stemmer*yang ditambahkan proses pengembalian akhiran. Proses pengembalian akhiran adalah proses yang dikembangkan dari metode *arifin stemmer* dimana pada proses tersebut dilakukan pengkombinasian sufik yang telah dihapus dengan kata dasar.

Karena metode *enhanced confix stripping stemmer* merupakan pengembangan dari metode *asian* dan *arifin stemmer,* metode tersebut mampu mengatasi beberapa kesalahan yang terjadi pada kedua metode pendahulunya. Untuk nilai *understemming* dan *overstemming,* metode *enhanced confix stripping*mendapatkan nilai 0,069 dan 0,001. Nilai tersebut merupakan yang paling rendah diantara metode-metode yang diuji.

Contoh dari keunggulan metode *enhanced confix stripping* terhadap metode *asian* misalnya pada kata *berarti* yang telah dibahas sebelumnya. Metode *asian* gagal mendapatkan kata dasar dari kata *berarti* karena tidak adanya aturan *rule precedence* untuk kombinasi imbuhan *be-i.* Tetapi metode *enhanced confix stripping* mampu mengatasi masalah tersebut dengan mengembalikan akhiran atau sufik terhadap kata dasar. Penjelasanya sebagai berikut, pertama metode *enhanced confix stripping* akan menghapus sufik *i* sehingga kata *berarti* berubah menjadi *berart.* Kemudian, penghapusan prefik dilakukan sehingga kata tersebut berubah menjadi *art.* Karena belum ditemukan pada kamus kata dasar, tahap *recoding* dilakukan sehingga kata berubah menjadi *rart.* Setelah sampai pada tahap recoding dan kata dasar masih belum ditemukan pada kamus, maka tahap pengembalian akhiran atau sufik dilakukan. Tahap tersebut mengubah kata pada tahap sebelum *recoding,* yaitu kata *berart* menjadi *berarti,* kemudian metode melakukan penghapusan prefik sehingga kata *berarti* menjadi *arti.* Kata *arti*

ditemukan pada kata dasar sehingga proses dihentikan dan kata *arti* adalah hasil akhir.

Dari penjabaran tersebut dapat disimpulkan bahwa dengan tahap pengembalian akhiran, metode *enhanced confix stripping stemmer* dapat menangani permasalahan *rule precedence* tanpa perlu mendefinisikan aturan baru, tetapi dengan tahap yang lebih panjang dibandingkan metode *asian.*

Untuk waktu pemrosesan data yang dibutuhkan metode *enhanced confix stripping stemmer* lebih lama dibandingkan dengan metode *asian stemmer*, yaitu 2701 detik untuk memproses data *twitter* dan 142 detik untuk memproses data kelompok kata. Waktu yanglama tersebut disebabkan adanya penambahan aturan pengembalian akhiran dan penambahan aturan pada penghapusan prefik.

Metode terakhir yang akan dibahas pada sub-bab 4.4.1 ini adalah metode *arifiyanti stemmer.* Metode tersebut merupakan pengembangan dari metode *asian stemmer* yang sengaja disesuaikan untuk melakukan *stemming* pada data media sosial khususnya data *twitter.*

Metode *arifiyanti stemmer* memperoleh nilai *understemming* sebesar0,086 dan 0,001 untuk nilai *overstemming.* Sedangkan untuk *stemming weight* diperoleh nilai 0,013. Dari hasil tersebut, dapat disimpulkan bahwa performa metode *arifiyanti* hampir sama dengan metode *asian stemmer.* Hal ini dapat disebabkan karena pada metode *arifiyanti* terdapat penambahan aturan-aturan yang disesuaikan dengan bahasa tidak baku yang umum di *twitter.* Tetapi pada data kelompok kata yang digunakan untuk pengujian performa metode *stemmer,* sebagian besar menggunakan kata baku. Hal tersebut menyebabkan aturan-aturan untuk bahasa tidak baku pada metode *arifiyanti stemmer* tidak berdampakbanyak pada data uji yang digunakan.

Namun demikian, terdapat kata tidak baku pada data yang berhasil diubah menjadi kata dasar oleh metode *arifiyanti*yaitu kata *diperbaikin.* Kombinasi imbuhan *di-in* termasuk aturan tambahan pada *rule precedence,* sehingga tahap penghapusan prefik dilakukan terlebih dahulu. Kata *diperbaikin* secara bertahap diubah menjadi *perbaikin* kemudian menjadi kata *baikin.* Setelah tahap tersebut, dilakukan penghapusan sufik *in* sehingga kata *baikin* berubah menjadi *baik.*

Karena kata *baik* ada pada kamus kata dasar, maka proses dihentikan dan kata *baik* dianggap hasil akhir.

Tetapi karena pada metode *arifiyanti stemmer* tidak terdapat proses mengembalikan akhiran seperti pada metode *enhanced confix stripping*, maupun proses pengkombinasian imbuhan dengan kata dasar seperti pada metode *arifin stemmer*, maka untuk kata seperti *berarti* dan *seharipun* tidak dapat diatasi.

Untuk waktu pemrosesan yang dibutuhkan oleh metode *arifiyanti* adalah 2416 detik untuk seluruh data *twitter* dan 124 detik untuk data kelompok kata. Waktu tersebut lebih lama jika dibandingkan dengan waktu metode *asian stemmer*, namun lebih cepat dari metode *arifin* dan *enhanced confix stripping stemmer*. Hal tersebut disebabkan karena penggunaan kamus kata tidak baku yang berisi sekitar 1000 kata pada metode *arifiyanti*. Namun, jumlah pengecekkan kamus kata dasar tidak sebanyak metode *arifin* dan *enhanced confix stripping*, karena pada metode *arifiyanti* tidak ada tahap tambahan seperti pengkombinasian imbuhan dan pengembalian sufik.

Dari hasil pengujian dan pembahasan, dapat disimpulkan bahwa performa terbaik didapat dengan metode *enhanced confix stripping stemmer* dengan nilai *understemming* 0,069. Metode tersebut mampu mengatasi kekurangan metode *asian stemmer* dengan tahap pengembalian akhiran yang dikembangkan dari metode *arifin stemmer*. Sedangkan performa yang paling buruk terjadi pada metode *porter confix stripping stemmer* dengan nilai *understemming* 0,227. Hal tersebut disebabkan oleh aturan yang didefinisikan ternyata tidak cocok untuk kata-kata bahasa Indonesia.

Untuk waktu proses tercepat dicapai dengan metode *fadillah stemmer* dengan waktu 0,627 detik. Hal tersebut disebabkan karena tidak digunakannya kamus kata dasar dan sebagai gantinya menggunakan tahap penghitungan suku kata. Sedangkan waktu proses terlama adalah metode *arifin stemmer* yaitu 3624 detik. Waktu yang lama tersebut disebabkan karena banyaknya jumlah pengecekkan kamus kata dasar bahasa Indonesia.

4.4.2 Analisis Hasil Performa Pengelompokan Data Dan Signifikansi Metode *Stemmer*

Pada sub-bab ini akan dibahas tentang hasil-hasil yang didapatkan pada pengujian performa pengelompokkan data menggunakan metode *davies-bouldin* (DBI) dan hasil pengujian signifikansi menggunakan uji *t.* Pembahasan dilakukan dalam satu sub-bab karena kedua nilai dari hasil pengujian saling berkaitan.

Pada Tabel 4.8 yang berisi nilai DBI pengelompokkan data, dapat disimpulkan bahwa nilai DBI terendah tidak konsisten pada satu metode *stemmer.* Hal tersebut disebabkan karena setiap pemilihan jumlah kelompok pada metode *k-means,* titik *centroid* akan dipilih secara random oleh metode.

Metode *porter confix stripping stemmer* memiliki rata-rata nilai DBI 9,214, yang lebih tinggi dibandingkan tanpa menggunakan *stemmer* yaitu 9,041. Sedangkan hasil uji *t* menunjukkan nilai signifikan. Artinya rata-rata performa pengelompokan data menjadi lebih buruk secara signifikan jika diterapkan metode *porter confix stripping stemmer.* Hal tersebut disebabkan karena metode *porter confix stripping* tidak men-*stemming* kata menjadi kata dasar namun terkadang malah mengubah kata dasar itu sendiri. Contohnya pada kata *protes* yang telah dibahas sebelumnya, metode *porter confix stripping* mengubahnya menjadi *prote.* Permasalahan tersebut akan menambah besar nilai *understemming.*

Untuk metode *nazief stemmer* didapatkan nilai rata-rata DBI 8,759, sedangkan hasil uji *t* terhadap nilai rata-rata tersebut menunjukkan signifikansi jika dibandingkan dengan tanpa penerapan *stemmer.* Hasil tersebut menunjukkan bahwa metode *nazief* cocok untuk diterapkan pada pengelompokan data teks *twitter.*

Metode *arifin stemmer* mendapatkan nilai rata-rata DBI 8,845, sedangkan hasil pengujian *t* tidak menunjukkan signifikansi dibandingkan tanpa penerapan *stemmer.* Hasil tersebut menunjukkan bahwa performa pengelompokan data dengan metode *arifin* masih dibawah metode *nazief stemmer.* Hal ini juga dikonfirmasi dengan lebih tingginya nilai *understemming* dari metode *arifin stemmer.*

Metode *fadillah stemmer* mendapatkan rata-rata nilai DBI8,926, sedangkan pengujian signifikansi tidak menunjukkan nilai signifikan jika

dibandingkan dengan tanpa menggunakan metode *stemmer*. Hal tersebut disebabkan karena kesalahan-kesalahan dari hasil *stemmer* yang telah dibahas sebelumnya sehingga nilai *understemming* lebih rendah dari metode *arifin stemmer*. Hal ini juga berakibat kurang bagusnya struktur *cluster* yang terbentuk sehingga nilai DBI juga lebih tinggi dari metode *arifin stemmer*.

Metode *asian stemmer* mendapatkan rata-rata nilai DBI 8,696, sedangkan hasil pengujian *t* menunjukkan adanya signifikansi terhadap nilai DBI tanpa penggunaan metode *stemmer*. Nilai tersebut juga lebih baik dibandingkan dengan metode *nazief stemmer*. Hal tersebut juga dikonfirmasi dengan nilai *understemming* metode *asian* yang lebih rendah dibandingkan metode *nazief stemmer*. Karena metode *asian* mampu men-*stemming* kelompok kata berimbuhan menjadi satu kata dasar, menyebabkan struktur *cluster* yang terbentuk lebih baik sehingga nilai DBI menjadi lebih rendah.

Metode *enhanced confix stripping stemmer* mendapat nilai rata-rata DBI 8,805, sedangkan hasil pengujian *t* menunjukkan tidak adanya signifikansi jika dibandingkan dengan rata-rata nilai DBI tanpa penggunaan metode *stemmer*. Nilai rata-rata DBI tersebut juga lebih tinggi daripada metode *asian stemmer*. Hal ini menjadi perhatian karena nilai *understemming* metode *enhanced confix stripping stemmer* adalah yang terendah. Ada tiga faktor yang menyebabkan hal ini terjadi. Faktor yang pertama adalah, adanya kesalahan *stemming* yang terjadi pada metode *enhanced confix stripping*. Contohnya pada kelompok kata *"mengerti pengertian"* diubah menjadi *"mengerti pengertian"*. Metode *enhanced confix stripping* berhasil dalam mengenali kata *mengerti* sebagai kata dasar, namun tidak untukkata*pengertian*. Faktor yang kedua adalah, metode *asian stemmer* ternyata mampu mengubah kelompok kata tersebut menjadi sama, yaitu *"erti erti"*. Meskipun kata *erti* bukan kata dasar yang diinginkan. Faktor yang ketiga adalah, data kelompok kata diambil berdasarkan sampel dari data *twitter*, sedangkan untuk pengujian pengelompokkan data dan perhitungan nilai DBI menggunakan data *twitter* keseluruhan. Faktor-faktor tersebut dapat menyebabkan nilai rata-rata DBI dari metode *asian* lebih rendah dari metode*enhanced confix stripping stemmer*.

Metode yang terakhir adalah *arifiyanti stemmer*. Metode tersebut mendapatkan nilai rata-rata DBI terendah yaitu 8,673, sedangkan hasil pengujian *t* menunjukkan adanya signifikansi dibandingkan rata-rata nilai DBI tanpa menggunakan *stemmer*. Hal ini dapat disebabkan karena aturan-aturan penghapusan imbuhan, baik prefik maupun sufik pada metode *arifiyanti* cocok dengan data *twitter* yang digunakan. Contohnya pada kata *diperbaikin* yang telah dibahas sebelumnya. Hal tersebut menyebabkan struktur *cluster* yang terbentuk lebih baik sehingga rata-rata nilai DBI menjadi lebih rendah.

Dari pembahasan yang telah dilakukan, dapat disimpulkan bahwa metode terbaik untuk melakukan pengelompokan data *twitter* secara umum adalah metode *arifiyanti*, diikuti dengan metode *asian* dan *nazief stemmer*.

Dari hasil dan analisis yang telah dibahas, dapat disimpulkan bahwa metode-metode *stemmer* secara umum akan memberikan peningkatan performa pengelompokan data. Namun, hal tersebut tidak terjadi pada metode *porter confix stripping stemmer*. Selain karena kesalahan-kesalahan hasil *stemmer* yang terjadi pada metode *porter*, juga disebabkan karena aturan-aturan yang tidak cocok dengan bahasa Indonesia. Jika dilihat dari nilai DBI pada Tabel 4.8 maka metode *stemmer* yang paling baik untuk meningkatkan performa pengelompokan data berturut-turut adalah metode*arifiyanti*, metode *asian*, metode *nazief*, metode *enhanced confix stripping*, metode*arifin*, dan terakhir metode *fadillah stemmer*.

4.4.3 Analisis Hubungan Performa Metode Stemmer Dengan Performa Pengelompokan Data

Metode *stemmer* yang baik seharusnya mampu men-*stemming* kata-kata yang berkolerasi menjadi satu kata dasar. Selain itu, metode *stemmer* yang baik juga harus mampu mencegah kata-kata yang tidak berkolerasi ter-*stemming* menjadi satu kata dasar. Oleh sebab itu, performa metode *stemmer* yang ditunjukkan dengan nilai *understemming*, *overstemming*, seharusnya memiliki dampak langsung pada performa *clustering* yang ditunjukkan dengan nilai DBI.

Untuk mengetahui hubungan antara performa metode *stemmer* dan performa *clustering*, penulis melakukan pengujian korelasi dengan metode

pearson (Pearson, 1895). Pengujian dilakukan dengan menggunakan perangkat lunak *R studio.* Hasil pengujian disajikan pada Tabel 4.10 berikut.

Tabel 4.10 Hasil Pengujian Korelasi Performa *Stemmer* Dan Performa *Clustering*

Korelasi	**df**	**Nilai korelasi**	**p-value**	**Kesimpulan**
Understemming dan DBI	5	-0,90091	0,00562	Berkorelasi negatif
Overstemming dan DBI	5	0.9354698	0,00196	Berkorelasi positif

Berdasarkan hasil pengujian korelasi pada Tabel 4.10, dapat disimpulkan bahwa nilai *overstemming* pada metode-metode *stemmer* bahasa Indonesia jika semakin rendah, maka nilai DBI akan menurun. Sebaliknya nilai *understemming* semakin randah, maka nilai DBI akan meningkat. Hasil pengujian juga membuktikan bahwa hasil penelitian Flores dan Moreira (2016) yang menyatakan bahwa nilai *overstemming* berkorelasi negatif dengan performa *information retrieval* tidak terjadi untuk bahasa Indonesia.

4.5 Permasalahan Penggunaan Kamus Kata Dasar Pada Metode *Stemmer*

Pada bagian ini akan dibahas permasalahan metode *stemmer* yang menggunakan kamus kata dasar bahasa Indonesia. Kamus kata dasar yang digunakan pada penelitian ini didapatkan dari :
http://hikaruyuuki.lecture.ub.ac.id/kamus-kata-dasar-dan-stopword-list-bahasa-indonesia.

Pada kamus yang berjumlah 28.500 kata tersebut ditemukan fakta bahwa huruf-huruf abjad yaitu *a, b, c* dan seterusnya termasuk anggota dari isi kamus tersebut. Hal ini dikonfirmasi dengan *Kamus Bahasa Indonesia* terbitan pusat bahasa departemen pendidikan nasional tahun 2008. Contoh dari definisi abjad *a* menurut *Kamus Bahasa Indonesia* adalah *huruf pertama abjad Indonesia, Ampere; lambang satuan ukuran arus listrik,* dan *are; nama satuan ukuran luas (=100m^2).* Huruf-huruf abjad tersebut memiliki simbol *n* pada *Kamus Bahasa Indonesia* yang artinya termasuk kata benda.

Fakta tersebut berpengaruh pada metode *stemmer* Indonesia, misalnya untuk metode *arifiyanti* dan *enhanced confix stripping stemmer*. Pada kedua metode tersebut sama-sama men-*stemming* kata *mengaku* menjadi *a*. Hal ini terjadi karena tahap penghapusan *possesive pronoun "ku"* yang menyebabkan kata *mengaku* menjadi *menga*. Karena belum ditemukan pada kamus, maka kedua metode manjalankan tahap penghapusan awalan/prefik, sehingga kata *menga* berubah menjadi *a*. Karena abjad *a* terdapat pada kamus, seluruh proses dihentikan dan *a* dianggap hasil akhir. Namun, abjad *a* bukan kata dasar yang diinginkan.

Permasalahan lainya yang timbul pada metode *arifiyanti stemmer* adalah untuk kata *rb*. Kata tersebut merupakan kata singkatan dari *ribu* yang sering digunakan pengguna *twitter*. Metode *arifiyanti* akan men-*stemming* kata *rb* menjadi *b*. Hal tersebut disebabkan adanya aturan tambahan yaitu "jika kata dimulai dengan *r*, maka coba hapus *r* dan cek kamus kata dasar". Karena abjad *b* memang ada pada kamus kata dasar, maka *b* dianggap kata dasar. Permasalahan yang sama juga timbul saat metode tersebut men-*stemming* kata *keb* menjadi *b*.

Kesalahan *stemming* yang terjadi pada kata *rb* dan *keb* dapat menimbulkan permasalahan *overstemming*. Hal tersebut disebabkan oleh kedua kata yang merupakan singkatan dan seharusnya tidak ter-*stemming*.

Walaupun demikian, pada penelitian ini huruf-huruf abjad tetap disertakan pada kamus kata dasar mengingat huruf-huruf tersebut memang valid pada *Kamus Bahasa Indonesia*.

4.6 Keterkaitan Hasil Pengujian Metode Stemmer Dengan Penelitian Terdahulu

Pada bagian ini akan dibahas mengenai keterkaitan dari penelitian terdahulu, yaitu Flores dan Moreira (2016), dengan hasil pengujian metode-metode stemmer untuk bahasa Indonesia.

Pada penelitian terdahulu didapatkan beberapa kesimpulan, diantaranya adalah metode *stemmer* yang akurat bukan merupakan satu-satunya cara untuk mendapatkan peningkatan performa *information retrieval,* namun metode *stemmer* yang sederhana juga mampu memberikan peningkatan performa.

Pernyataan tersebut ternyata hanya sebagian yang benar, jika melihat dari hasil pengujian yang telah dilakukan. Pernyataan bahwa metode *stemmer* yang akurat bukan merupakan satu-satunya cara untuk mendapatkan peningkatan performa *information retrieval* adalah benar jika melihat pada metode *enhanced confix stripping*. Pada Tabel 4.6, metode tersebut terbukti akurat, namun rata-rata nilai DBI yang didapatkan tidak signifikan. Sementara metode *arifiyanti* mendapatkan rata-rata nilai DBI yang paling baik. Hal ini dibuktikan juga pada metode modifikasi *arifiyanti* yang akan dibahas pada sub-bab selanjutnya.

Namun, pernyataan bahwa metode yang sederhana juga mampu memberikan peningkatan yang baik pada performa *information retrieval* tidak terjadi pada pengujian ini.Contoh dari metode sederhana pada penelitian ini adalah metode *fadillah stemmer* yang hanya menggunakan aturan penghapusan imbuhan, tanpa adanya proses seperti *recoding, rule precedence,* dan lainya. Jika dilihat pada Tabel 4.6, metode ini mendapat nilai kesalahan *understemming* yang tinggi yaitu 0,1322, sementara kesalahan *overstemming* yang hambir sama denganmetode lainya yaitu 0,0011. Jika dilihat pada Tabel 4.8, maka metode *fadillah* mendapatkan performa *clustering* paling rendah setelah metode *porter confix stripping*dengan nilai rata-rata DBI pada Tabel 4.8 yaitu 8,926. Hal serupa juga terjadi pada metode *porter confix stripping.*Dari hasil tersebut dapat disimpulkan bahwa semakin rendah nilai kesalahan *understemming* dan *overstemming* maka performa *clustering* menjadi lebih baik.

Faktor yang dapat menyebabkan perbedaan kesimpulan tersebut diantaranya adalah struktur bahasa Indonesia yang berbeda dengan bahasa-bahasa yang diuji pada penelitian terdahulu (Flores dan Moreira, 2016). Contohnya adalah aturan *recoding* pada beberapa metode *stemmer nazief*yang sengaja dibuat karena adanya peleburan imbuhan pada bahasa Indonesia. Contoh dari peleburan tersebut diantaranya imbuhan *pe-*, dan *me-* yang dapat berubah menjadi *peny-*, dan *meny-*. Hal tersebut tidak terdapat pada metode seperti *porter confix stripping stemmer.*

4.7 Perbandingan Metode *Stemmer* Dengan Metode Weighted Scoring

Pembahasan pada sub-bab sebelumnya membandingkan metode-metode *stemmer* dari satu sudut pandang saja, misalnya dari sisi performa *clustering* atau dari sisi waktu pemrosesan data. Oleh sebab itu, pada sub-bab ini dilakukan perbandingan menggunakan metode *weighted scoring* agar mendapat gambaran performa metode *stemmer* yang lebih menyeluruh. Hasil perbandingan ini akan digunakan untuk merumuskan kontribusi praktis pada Bab selanjutnya.

4.7.1 Hasil Normalisasi Nilai Pada Kriteria Metode-Metode *Stemmer*

Proses normalisasi dilakukan untuk mengeliminasi unit pengukuran yang berbeda diantara nilai DBI, nilai *understemming*, nilai *overstemming*, dan waktu proses. Pada Tabel 4.11 berikut disajikan nilai dari kriteria metode *stemmer* sebelum tahap normalisasi, sedangkan Tabel 4.12 adalah nilai ternormalisasi.

Tabel 4.11Nilai Kriteria Metode *Stemmer*

Metode *Stemmer*	Kriteria			
	DBI	Understem-ming	Overstem-ming	Waktu Proses
Porter Confix Stripping	9,21400	0,22662	0,00000	0,62700
Nazief	8,75900	0,10417	0,00112	1961,87900
Arifin	8,84500	0,11097	0,00110	3624,80400
Fadillah	8,92600	0,13223	0,00111	0,21800
Asian	8,69600	0,08759	0,00115	1967,30100
Enhanced Confix Stripping	8,80500	0,06845	0,00117	2702,45000
Arifiyanti	8,67300	0,08631	0,00117	2416,49900

Tabel 4.12Nilai Kriteria Metode *Stemmer* Ternormalisasi

Metode *Stemmer*	Kriteria			
	DBI	Understem-ming	Overstem-ming	Waktu Proses
Porter Confix Stripping	0,00000	0,00000	1,00000	0,99989
Nazief	0,84104	0,77419	0,03602	0,45879
Arifin	0,68207	0,73118	0,05403	0,00000
Fadillah	0,53235	0,59677	0,04803	1,00000
Asian	0,95749	0,87903	0,01201	0,45729
Enhanced Confix Stripping	0,75601	1,00000	0,00000	0,25447
Arifiyanti	1,00000	0,88710	0,00000	0,33336

4.7.2 Hasil Perbandingan Metode *Stemmer* Dengan Metode Weighted Scoring

Setelah melakukan tahap normalisasi nilai pada kriteria-kriteria metode *stemmer*, dilakukan perhitungan skor dengan metode *weighted scoring*. Hasil perhitungan skor disajikan pada Tabel 4.13. Warna merah menunjukkan skor terendah, sedangkan warna kuning menunjukkan skor tertinggi.

Tabel 4.13 Hasil Perhitungan Dengan Metode *Weighted Scoring* Untuk Metode *Stemmer*

Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Porter Confix Stripping	Rata-rata DBI	0,45	0,00000	0,00000
	Understemming	0,20	0,00000	0,00000
	Overstemming	0,25	1,00000	0,25000
	Waktu Proses	0,10	0,99989	0,09999
Total Nilai :				0,34999
Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Nazief	Rata-rata DBI	0,45	0,84104	0,37847
	Understemming	0,20	0,77419	0,15484
	Overstemming	0,25	0,03602	0,00901
	Waktu Proses	0,10	0,45879	0,04588
Total Nilai :				0,58819
Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Arifin	Rata-rata DBI	0,45	0,68207	0,30693
	Understemming	0,20	0,73118	0,14624
	Overstemming	0,25	0,05403	0,01351
	Waktu Proses	0,10	0,00000	0,00000
Total Nilai :				0,46668
Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Fadillah	Rata-rata DBI	0,45	0,53235	0,23956
	Understemming	0,20	0,59677	0,11935
	Overstemming	0,25	0,04803	0,01201
	Waktu Proses	0,10	1,00000	0,10000
Total Nilai :				0,47092
Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Asian	Rata-rata DBI	0,45	0,95749	0,43087
	Understemming	0,20	0,87903	0,17581
	Overstemming	0,25	0,01201	0,00300
	Waktu Proses	0,10	0,45729	0,04573
Total Nilai :				0,65541

Tabel 4.13 Hasil Perhitungan Dengan Metode *Weighted Scoring* Untuk Metode *Stemmer*(Lanjutan)

Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Enhanced Confix Stripping	Rata-rata DBI	0,45	0,75601	0,34020
	Understemming	0,20	1,00000	0,20000
	Overstemming	0,25	0,00000	0,00000
	Waktu Proses	0,10	0,25447	0,02545
Total Nilai :				0,56565
Metode Stemmer	**Kriteria**	**Bobot**	**Skor**	**Nilai**
Arifiyanti	Rata-rata DBI	0,45	1,00000	0,45000
	Understemming	0,20	0,88710	0,17742
	Overstemming	0,25	0,00000	0,00000
	Waktu Proses	0,10	0,33336	0,03334
Total Nilai :				0,66076

Dari hasil perhitungan skor untuk metode-metode *stemmer*, metode *arifiyanti* mendapatkan nilai skor tertinggi yaitu 0,660. Hal tersebut disebabkan oleh nilai DBI nya yang rendah dan performa *understemming*-nya yang juga baik. Pada urutan kedua adalah metode *asian*dengan skor 0,655 yang juga memiliki nilai DBI rendah dan nilai *understemming* yang juga rendah.

Pada urutan ketiga adalah metode *nazief*(0,588) diikuti metode *enhanced confix stripping* (0,565), *fadillah*(0,470), *arifin* (0,467). Pada perhitungan skor ini, metode *fadillah* lebih unggul dari metode *arifin stemmer* karena nilai *overstemming* yang sangat baik dan waktu proses yang sangat cepat. Sedangkan metode *arifin stemmer*, walaupun performa *clusteringnya* baik, waktu prosesnya adalah yang terlama dibandingkan metode-metode lainnya.

Pada urutan terakhir adalah metode *porter confix stripping*. Metode ini memiliki nilai *overstemming* yang sangat baik, tetapi performa *clustering* dan nilai *understemming* sangat buruk dibandingkan metode-metode lainnya.

BAB 5
KONTRIBUSI PENELITIAN

Pada sub-bab ini akan dibahas kontribusi dari penelitian yang telah dilakukan. Kontribusi yang disajikan meliputi kontribusi keilmuan dan kontribusi praktis.

5.1 Kontribusi Keilmuan

Kontribusi keilmuan didapatkan setelah melakukan tahap pengujian dan analisis hasil. Metode *arifiyanti stemmer* yang merupakan metode paling baru yang diketahui penulis masih memiliki beberapa kekurangan. Hal tersebut dapat disebabkan karena metode ini dikembangkan berdasarkan metode *asian* sehingga tambahan aturan yang ada pada metode *enhanced confix stripping* terlewatkan.

Oleh sebab itu, penulis menambahkan aturan dari tahap penghapusan prefik yang ada dalam metode *enhanced confix stripping* pada metode *arifiyanti stemmer*. Aturan yang ditambahkan meliputi penghapusan untuk *mem+p…, men+s…, menge+…, penge+…, peng+k…*. Selain itu, penulis mencoba menambahkan tahap *stemming ruleprecedence* pada tahap akhir *stemming non-ruleprecedence*. Sehingga pada saat proses *recoding* gagal menemukan kata dasar, maka langsung lakukan tahap *stemming ruleprecedence*, dan bila masih belum ditemukan pada kata dasar, barulah kata masukkan dianggap kata dasar. Penulis memberikan label *arifiyanti+aturanECS* untuk metode tersebut.

Selain itu, penulis juga mencoba dengan hanya menambahkan aturan pengecekkan *ruleprecedence*. Adapun aturan *ruleprecedence* yang ditambahkan pada metode *arifiyanti* adalah *be-i, di-pun, se-pun, s-an, di-nya,* dan *te-ah*. Metode ini dilabeli oleh penulis dengan *arifiyanti+aturanbaruRP*.

Kedua metode tersebut diuji dengan data dan tahap pengujian yang sama dengan metode-metode *stemmer* yang ada sebelumnya, dan hasilnya sebagai berikut :

Tabel 5.1 Indeks *Understemming* dan *Overstemming* Metode Modifikasi

Nomor	Metode Stemmer	*Overstemming Index*	*Understemming Index*	*Stemming Weight*
1	*Arifiyanti Stemmer*	0,001166	0,086309	0,013509
2	*Arifiyanti+aturanECS*	0,001186	0,072279	0,016418
3	*Arifiyanti+ aturanbaruRP*	0,001172	0,083333	0,014075

Tabel 5.2Waktu Pengujian Metode *Stemmer*Modifikasi

Nama Metode	Waktu Proses (Detik)
Arifiyanti Stemmer	2416,499
Arifiyanti+aturanECS	5318,764
Arifiyanti+ aturanbaruRP	2995,695

Dari hasil pengujian pada Tabel 5.1, dapat disimpulkan bahwa keakuratan metode *arifiyanti* meningkat pada kedua metode modifikasi. Peningkatan terbesar pada metode *arifiyanti+aturanECS* didapatkan karena proses pengecekkan *ruleprecedence* apabila langkah *non-ruleprecedence* gagal. Langkah ini berfungsi seperti *loopPengembalianAkhiran* pada metode *enhanced confix stripping*. Namun penambahan tahap *stemming ruleprecedence* membuat waktu proses bertambah sebab proses tersebut menyebabkan dua kali tahap *stemming* dilakukan pada satu kata.

Selanjutnya adalah hasil pengujian DBI pada *clustering* akan disajikan pada Tabel 5.3 dan Tabel 5.4 berikut.

Tabel 5.3Hasil Perhitungan DBI Pengelompokan Data

Kelompok	*Arifiyanti*	*Arifiyanti+aturanECS*	*Arifiyanti+aturanbaruRP*
2	**12.926**	14.257	12.927
3	**10.601**	12.625	11.432
4	11.696	11.600	**10.378**
5	10.159	**9.307**	10.026
6	9.615	9.602	**9.592**
7	9.733	**8.862**	9.582
8	**8.878**	8.934	8.950
9	8.361	**8.339**	8.584
10	8.561	8.265	**7.943**

Kelompok	*Arifiyanti*	*Arifiyanti+aturanECS*	*Arifiyanti+aturanbaruRP*
11	**7.693**	8.445	8.309
12	8.366	**7.779**	8.120
13	7.635	**7.586**	7.655
14	7.526	**7.247**	7.348
15	7.749	**7.437**	7.652
16	**6.885**	7.332	7.538
17	7.397	7.147	**6.773**
18	7.438	**6.760**	6.771
19	**6.845**	7.069	7.176
20	**6.717**	7.030	6.992
Rata-rata	8.673	8.717	**8.618**

Tabel 5.4Hasil Pengujian *t* Pada Metode Modifikasi

	Tanpa Stemmer	*Arifi-yanti*	*Enhanced Confix Stripping*	*Asian*	*Fadi-llah*	*Arifin*	*Nazief*	*Porter confix stripping*	*Arifiyanti+aturanECS*
Arifiyanti+aturanECS	=	=	=	=	=	=	=	+	
Arifiyanti+aturan baruRP	+	=	=	=	+	=	=	+	=

Dari hasil pengujian DBI terlihat bahwa metode *arifiyanti+aturanECS* memiliki rata-rata DBI kurang dari metode *arifiyanti*. Hal tersebut dapat disebabkan karena penambahan aturan dari metode *enhanced confix stripping* menyebabkan metode tidak lagi men-*stemming* kata *pengertian* menjadi *erti*. Hasil tersebut juga mengkonfirmasi hasil penelitian Flores dan Moreira (2016) bahwa untuk mendapatkan peningkatan performa pada *information retrieval* suatu metode *stemmer* tidak harus sangat akurat. Sebaliknya, penambahan aturan baru pada metode *arifiyanti+aturanbaruRP* mampu meningkatkan performa keakuratan *arifiyanti stemmer* walau tidak seakurat metode *arifiyanti+aturanECS*. Selain itu, performa *clustering* juga mengalami peningkatan walaupun peningkatan tersebut ternyata tidak signifikan dibandingkan metode *arifiyanti*. Tidak signifikannya perbedaan diantara ketiga metode tersebut dapat disebabkan karena jumlah kata-kata yang memenuhi aturan baru pada metode modifikasi tidak cukup banyak dalam data yang diuji.

5.2 Kontribusi Praktis

Kontribusi praktis dari penelitian ini adalah metode *stemmer* yang paling baik untuk diterapkan pada pengelompokan data keluhan pelanggan PLN.

Berdasarkan hasil perbandingan metode-metode *stemmer* dengan *weighted scoring* pada Tabel 4.11, maka disarankan metode *stemmer* yang paling cocok diterapkan pada pengelompokan data keluhan pelanggan PLN adalah metode *arifiyanti stemmer*. Hal ini disimpulkan karena beberapa pertimbangan, yang pertama karena performa *clustering* yang sangat baik dengan skor 1,00. Pertimbangan yang kedua karena performa metode *stemmer*-nya yang juga baik dengan skor 0.887 untuk nilai *understemming*.

Dengan adanya batas waktu untuk pengambilan data, maka disarankan untuk melakukan pengumpulan data setiap 9 hari.

Selain kontribusi dari hasil pengujian dan analisis data yang telah dilakukan, penulis juga merancang sistem pengelompokan keluhan pelanggan untuk PT. PLN yang akan disajikan pada Sub-bab 5.2.1 dan Sub-bab 5.2.2 berikut.

5.2.1 Sistem Pengelompokan Keluhan Pelanggan Untuk PT. PLN

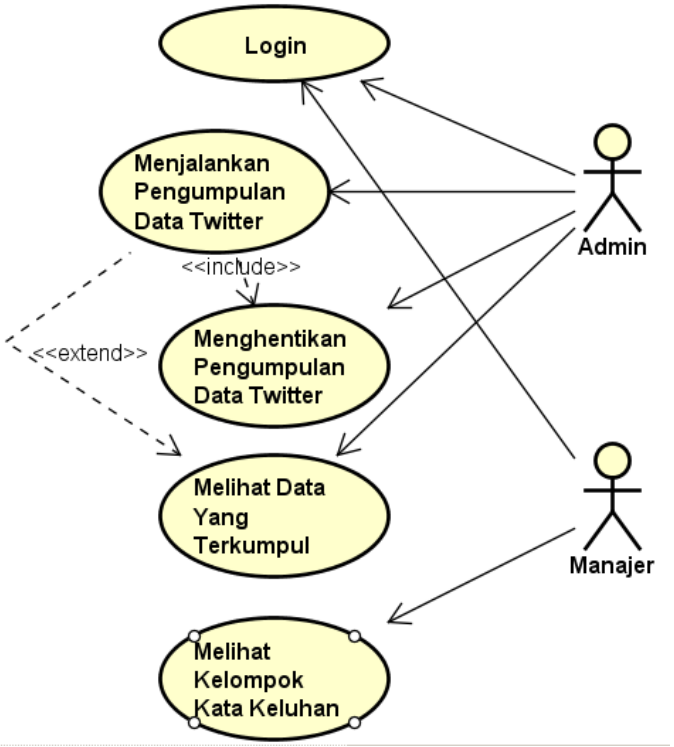
Pada bagian ini akan dijelaskan mengenai rancangan sistem untuk pengelompokan keluhan pelanggan pada PLN. Sistem digambarkan dengan beberapa diagram, diantaranya *use case* dan *sequence* diagram. Berikut ini adalah diagram *use case* untuk sistem yang akan dibuat.

Tabel 5.5*Use Case*

Nomor	Use Case	Aktor	Keterangan
1	Login	Admin Manajer	-admin dan manajer melakukan login ke dalam sistem untuk medapatkan hak akses masing-masing.
2	Menjalankan pengumpulan data twitter	Admin	-admin menjalankan perintah agar sistem melakukan pengumpulan data

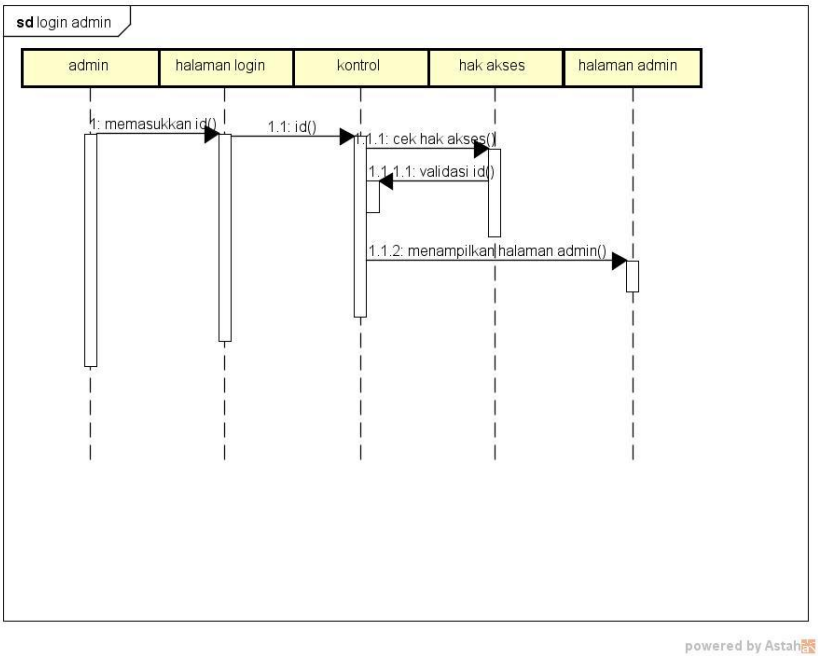
Tabel 5.5*Use Case*(Lanjutan)

Nomor	Use Case	Aktor	Keterangan
3	Menghentikan pengumpulan data twitter	Admin	-admin menjalankan perintah untuk menghentikan pengumpulan data
4	Melihat data yang terkumpul	Admin	-admin melihat data twitter yang telah dikumpulkan
5	Melihat kelompok kata keluhan	Manajer	-manajer melihat kelompok-kelompok kata dari data yang dikumpulkan



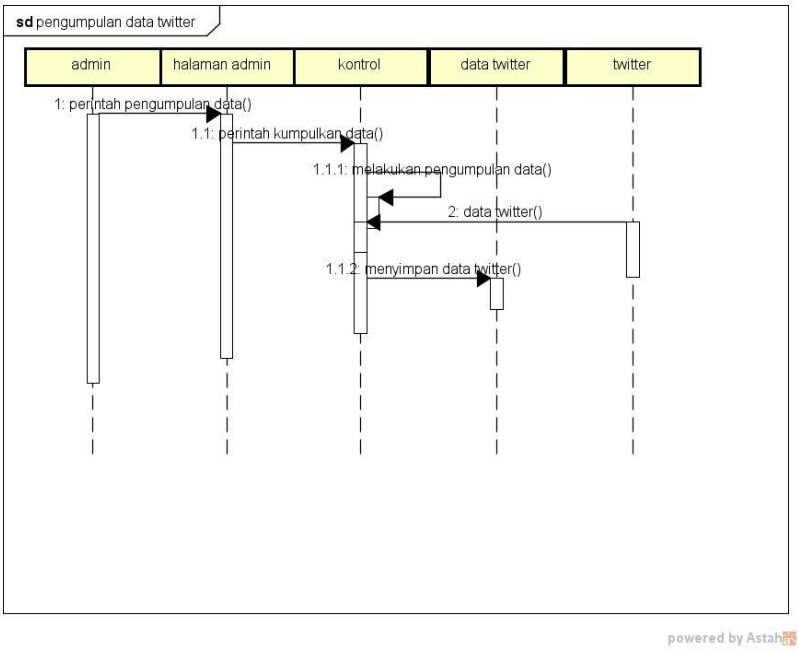
Gambar 5.1 *Use Case*

Setelah melakukan pemodelan sistem menggunakan *use case diagram*, maka setiap *use case* akan dibuat menjadi *sequence diagram. Sequene diagram* akan menjelaskan proses yang terjadi pada sistem dengan lebih detail.



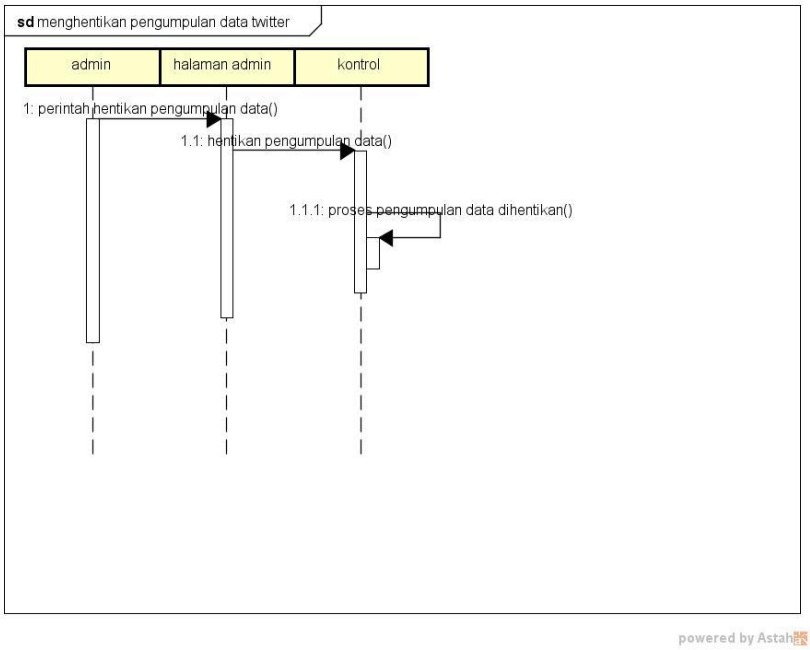
Gambar 5.2 Proses Login

Gambar 5.2 menggambarkan proses login dari admin, sedangkan untuk manajer juga menggunakan proses yang sama.



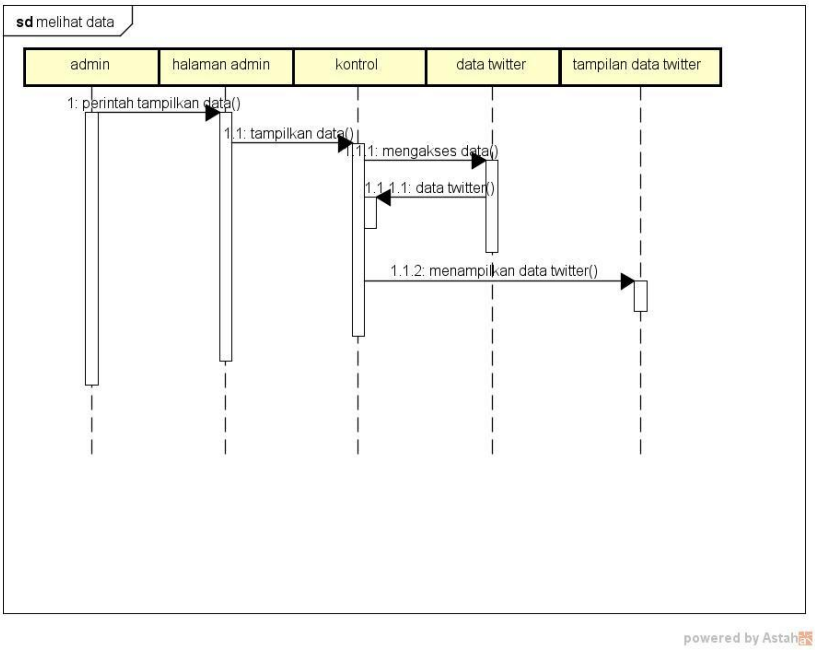
Gambar 5.3Proses Pengumpulan Data Twitter

Gambar 5.3 menjelaskan proses pengumpulan data *twitter* yang dilakukan oleh admin.



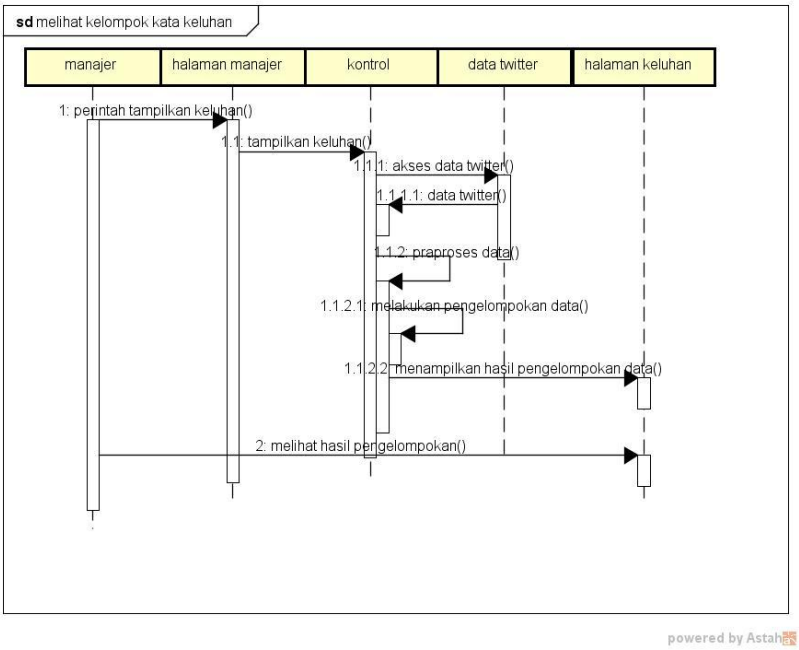
Gambar 5.4Proses Menghentikan Pengumpulan Data Twitter

Gambar 5.4 menjelaskan proses menghentikan pengumpulan data *twitter* yang dilakukan oleh admin.



Gambar 5.5Proses Melihat Data Twitter

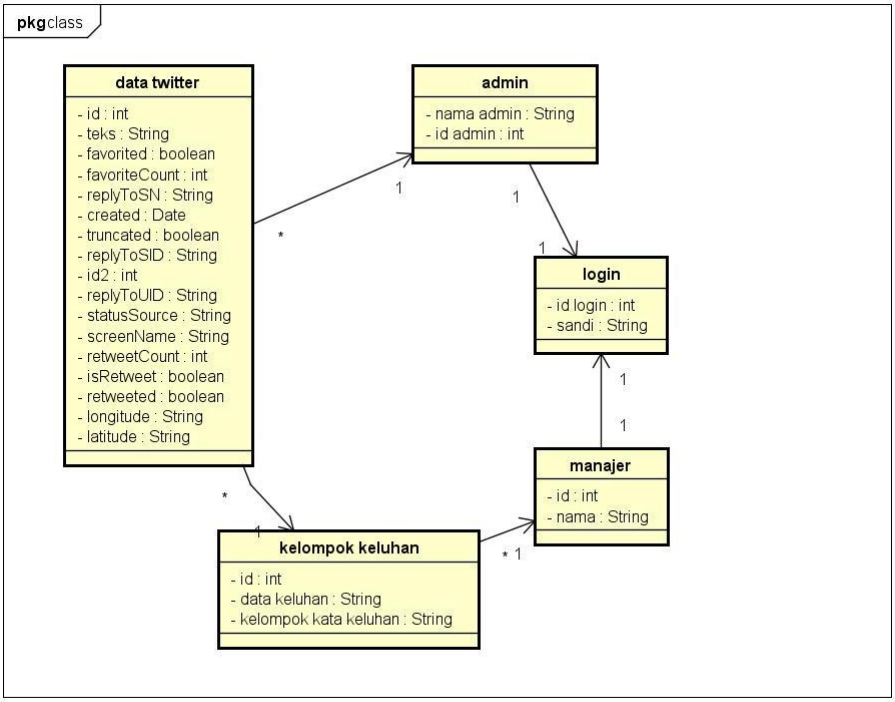
Gambar 5.5 menjelaskan proses untuk melihat data *twitter* yang telah terkumpul oleh admin.



Gambar 5.6Proses Melihat Kelompok Keluhan

Gambar 5.6 menjelaskan proses untuk melihat kelompok data twitter yang sudah diproses. Pada tahap ini, manajer melihat dan menginterpretasi keluhan yang ada.

Selain menjabarkan sistem dengan *use case* dan *sequence diagram*, penulis juga merancang *class diagram* yang akan menjelaskan atribut dari objek dalam sistem sebagai berikut.

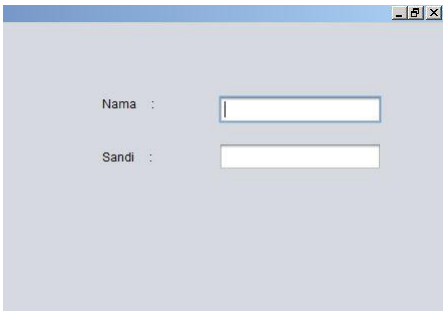


Gambar 5.7Diagram Kelas Dari Sistem Pengelompokan Data Keluhan Pelanggan PLN.

Gambar 5.7 menjelaskan tentang diagram kelas dari sistem pengelompokan keluhan pelanggan PLN. Pada diagram tersebut, setiap data hanya memiliki satu kelompok keluhan dan satu kelompok keluhan terdiri dari sejumlah data twitter. Atribut utama dari data *twitter* yang digunakan oleh sistem adalah *teks* yang berisi komentar pelanggan.

5.2.2 Rancangan Tampilan Sistem Pengelompokan Keluhan Pelanggan Untuk PT. PLN

Pada bagian ini akan dijelaskan tentang rancangan tampilan dari sistem yang akan dibuat. Rancangan tampilan dibuat berdasarkan *sequence diagram* yang telah dijelaskan pada Bab 5.2.1.

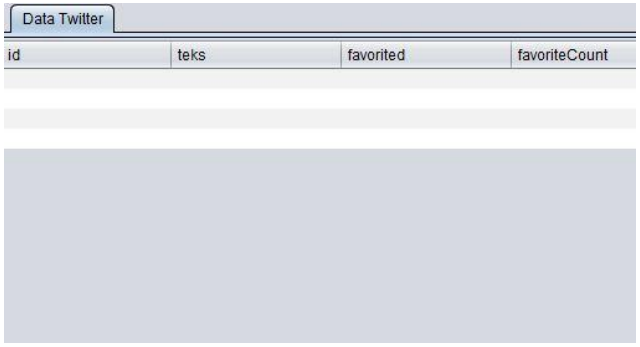


Gambar 5.8 Log In

Gambar 5.8 digunakan untuk *log in* sistem oleh admin selain itu, manajer juga menggunakan halaman *log in* serupa.

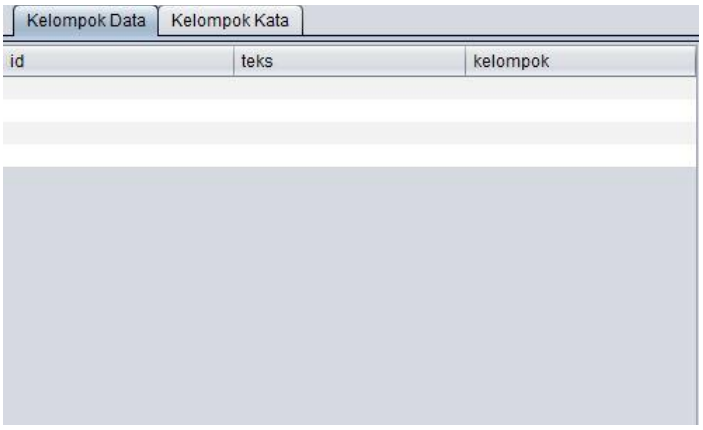
Gambar 5.9 Halaman Admin

Gambar 5.9 digunakan admin untuk memulai proses pengumpulan data dari *twitter.*



Gambar 5.10 Tampilan Data Twitter

Gambar 5.10 digunakan oleh admin untuk melihat data *twitter* yang telah dikumpulkan sebelumnya.



Gambar 5.11 Tampilan Kelompok Data

Gambar 5.11 digunakan oleh manajer untuk melihat data *twitter* yang telah dikelompokkan.

Gambar 5.12 Tampilan Kelompok Kata

Gambar 5.12 digunakan oleh manajer untuk melihat data *twitter* yang telah dikelompokkan menjadi kata unik. Daftar kata-kata tersebut yang akan digunakan untuk menginterpretasi keluhan pelanggan yang ada pada *twitter*.

Halaman ini sengaja dikosongkan

BAB 6
KESIMPULAN DAN SARAN

Bab terakhir ini terdiri atas kesimpulan dan saran-saran yang disajikan sebagai berikut.

6.1 Kesimpulan

Pada sub-bab ini akan dibahas mengenai kesimpulan yang didapatkan dari penelitian tentang perbandigan *stemmer* bahasa Indonesia dan dampaknya pada penggalian teks bahasa Indonesia, untuk studi kasus pengelompokan keluhan pelanggan PLN. Kesimpulan yang didapatkan diantaranya :

1. Untuk melakukan penerapan metode-metode *stemmer* bahasa Indonesia, pada data twitter yang telah didapatkan, dilakukan tahapan berikut:
    1) Mengumpulkan data dari varibel *text* menjadi satu dokumen.
    2) Melakukan penghapusan *retweet.*
    3) Melakukan tahap *tokenizing*, *case folding, stopwords removal, penghapusan karakter berulang*, dan *stemming.*
2. Secara umum metode *stemmer* yang paling baik untuk meningkatkan performa *information retrieval*, dalam penelitian ini kasus pengelompokan data, secara berturut-turut adalah metode *arifiyanti* dengan rata-rata nilai DBI=8,673, metode *asian*(8,696), metode *nazief* (8,759), metode *enhanced confix stripping*(8,805), metode *arifin*(8,845), dan terakhir metode *fadillah stemmer*(8,926).
3. Berdasarkan pengujian korelasi, nilai *overstemming* berkorelasi positif signifikan terhadap performa *clustering.* Sebaliknya, nilai *understemming* berkorelasi negatif signifikan.
4. Metode *stemmer* yang paling tepat diterapkan untuk pengelompokan data keluhan pelanggan dari *twitter* PLN adalah metode *arifiyanti stemmer.* Hal tersebut berdasarkan pemilihan menggunakan *weighted scoring*, dan metode *arifiyanti* mendapat skor tertinggi yaitu 0,661.
5. Metode *stemmer* yang memiliki rata-rata nilai DBI terendah adalah *arifiyanti stemmer*yaitu 8,673. Hal tersebut disebabkan oleh metode *arifiyanti* memiliki aturan-aturan yang sengaja dibuat untuk menyesuaikan

dengan data *twitter*. Contohnya aturan *di-in* yang berguna untuk men-*stemming* kata *dibalikin.*

6. Metode *stemmer* yang akurat untuk kata baku adalah *enhanced confix stripping stemmer*. Hal tersebut didukung oleh nilai indeks *understemming* yang rendah, yaitu 0,069.
7. Namun demikian, kamus kata dasar yang digunakan oleh metode-metode *stemmer* tersebut terdapat huruf-huruf abjad seperti *a, b, c,* dan seterusnya. Hal tersebut dapat menyebabkan metode *stemmer* salah dalam men-*stemming* kata.
8. Hasil yang didapat dari penelitian Flores dan Moreira (2016) ternyata tidak sepenuhnya cocok dengan yang terjadi pada bahasa Indonesia. Metode *fadillah stemmer* yang tidak menggunakan kamus kata dasar, ternyata tidak mendapat performa yang baik untuk nilai *understemming* maupun rata-rata nilai DBI hasil *clustering.* Selain itu, korelasi antara *understemming, overstemming* dengan *information retrieval* juga tidak cocok dengan korelasi yang terjadi pada bahasa Indonesia. Hal tersebut dapat disebabkan oleh struktur bahasa Indonesia yang berbeda dengan bahasa-bahasa yang diuji pada penelitian terdahulu. Contohnya terdapat peleburan imbuhan *pe-*, dan *me-* yang dapat berubah menjadi *peny-*, dan *meny-*.
9. Penambahan aturan untuk metode *arifiyanti* dapat meningkatkan keakuratan metode tersebut, dengan penurunan nilai *understemming* dari 0,086 menjadi 0,083 (*arifiyanti+ aturanbaruRP*) dan 0,072 (*arifiyanti+aturanECS*). Namun perbedaan performa *clustering* tidak signifikan.

6.2 Saran

Pada sub-bab ini akan dibahas mengenai saran yang didapatkan dari penelitian tentang perbandigan *stemmer* bahasa Indonesia dan dampaknya pada penggalian teks bahasa Indonesia sebagai berikut.

Pada penelitian ini digunakan kamus kata dasar yang memiliki huruf-huruf abjad (*a, b, c,* dan seterusnya) yang dapat menyebabkan metode *stemmer*

mengalami kesalahan saat melakukan pengecekkan kata dasar, misalnya kata *mengaku* menjadi *a* seperti pembahasan pada Sub-bab 4.5. Belum diketahui bagaimana dampak yang terjadi pada metode *stemmer* apabila huruf-huruf tersebut tidak digunakan sebagai kata dasar. Oleh sebab itu, disarankan melakukan *stemming* dengan tidak menyertakan huruf-huruf abjad tersebut pada kamus kata dasar.

Halaman ini sengaja dikosongkan

DAFTAR RUJUKAN

Abdi, H. 2010. *Normalizing Data-Encyclopedia of Research Design.* Thousand Oaks-California. Sage.

Achimugu, P.; Selamat, A.; Ibrahim, R.; dan Mahrin, M.N. 2014.A Systematic Literature Review of Software Requirements Prioritization Research.*Information and Software Technology 56,* 568–585.

Agusta, Y. 2007. *K-means*: Penerapan, Permasalahan dan Metode Terkait. *Jurnal Sistem dan Informatika 3,* 47-60.

Alam, M.J; Oullet, P.; Kenny, P.; O'Saughnessy, D. 2011. Comparative Evaluation of Feature Normalization Techniques for Speaker Verification. *Advances in Nonlinear Speech Processing,* 246-253.

Ali, I.; Guo, Y.; Silins, I.; Högberg, J.; Stenius, U.; dan Korhonen, A. 2016.Grouping Chemicals for Health Risk Assessment: A Text Mining-based Case Study of Polychlorinated Biphenyls (PCBs).*Toxicology Letters 241,* 32–37.

Arifin, A.Z. dan Setiono, A.N. 2002. *Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering*. Prosiding Seminar onIntelligent Technology and Its Applications (SITIA). Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Arifin, A.Z.; Mahendra, IP.A.K.; dan Ciptaningtyas, H.T. 2009. *Enhanched Confix Stripping Stemmer and Ants Algorithm for Classifying News Document in Indonesian.* International Conference on Information & Communication Technology and Systems.

Arifiyanti, A.A. 2015.*Ekstraksi Fitur pada Konten Jejaring Sosial Twitter Berbahasa Indonesia dalam Peningkatan Kinerja Klasifikasi Sentimen*, Master Thesis. Sistem Informasi, Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya.

Asian, J. 2007.*Effective Techniques for Indonesian Text Retrieval*.Thesis. School of Computer Science and Information Technology,Science, Engineering, and Technology Portfolio,RMIT University Melbourne.

Bakar, N.H.; Kasirun, Z.M.; dan Salleh, N. 2015.Feature Extraction Approaches from Natural Language Requirements for Reuse in Software Product Lines: A Systematic Literature Review.*The Journal of Systems and Software 106,* 132–149.

Bhanuse, S.S.; Kamble, S.D.; dan Kakde, S.M. 2016.*Text Mining Using Metadata for Generation of Side Information.* International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015,Nagpur, INDIA

Boukhari, K.; Omri, M.N. 2015. *SAID : A new Stemmer Algorithm to Indexing Unstructured Document*. 15th International Conference on Intelligent Systems Design and Applications. Tunisia

Brychcín, T. dan Konopík, M. 2015. HPS: High Precision Stemmer.*Information Processing and Management 51,* 68-91.

Cambra-Fierro, J.; Melero, I.; dan Sese, F.J. 2015.Managing Complaints to Improve Customer Profitability.*Journal of Retailing 91*,109–124.

Charalampakis, B.; Spathis, D.; Kouslis, E.; dan Kermanidis, K. 2016.A Comparison between Semi-supervised and Supervised Text Mining Techniques on Detecting Irony in Greek Political Tweets.*Engineering Applications of Artificial Intelligence 51,* 50-57.

Coussement, K. dan den Poel, D.V. 2008.Improving Customer Complaint Management by Automatic Email Classification Using Linguistic Style Features as Predictors.*Decision Support Systems 44*, 870-882.

Dalwadi, B.; Desai, N. 2016. *An Affix Removal Stemmer for Gujarati Text*.International Conference on Computing for Sustainable Global Development. India.

David, M.W. 2011.Evaluation: From Precision, Recall, and F-Measure to ROC, Informadeness, Markedness, & Correlation.*Journal of Machine Learning Technologies 2(1),* 37-63.

Dong, S. dan Wang, Z. 2015.*Evaluating Service Quality in Insurance Customer Complaint Handling throught Text Categorization,* 978-1-4799-1891-1/15 IEEE.

Ediyanto; Mara, M.N.; dan Satyahadewi, N. 2013. *Pengklasifikasian Karakteristik dengan Metode K-means Cluster Analysis.Buletin Ilmiah Mat. Stat. dan Terapannya (Bimaster)* 2(2), 133-136.

Faed, A.; Chang, E.; Saberi, M.; Hussain, O.K.; dan Azadeh, A. 2015, Intelligent Customer Complaint Handling Utilising Principal Component and Data Envelopment Analysis (PDA), *Applied Soft Computing*,

Fattah, M.A. 2015.New Term Weighting Schemes with Combination of Multiple Classifiers for Sentiment Analysis.*Neurocomputing 167,* 434-442.

Feldman, R dan Sanger, J. 2007. *The Text Mining Hand Book.* Cambridge. Cambridge University Press.

Flores,F.N. danMoreira, V.P. 2016. Assessingthe ImpactofStemmingAccuracyonInformationRetrieval: A Multilingual Perspective.*Information Processing and Management 52,* 840-854.

Galitsky, B.A.; González, M.P.; dan Chesñevar, C.I. 2009.A Novel Approach for Classifying Customer Complaints through Graphs Similarities in Argumentative Dialogues.*Decision Support Systems 46,* 717-729.

Gupta, V.; Joshi, N.; Mathur, I. 2013. *Rule Based Stemmer in Urdu.* 4th International Conference on Computer and Communication Technology. India

Han, J. dan Kamber,M. 2006. *Data Mining Concepts and Techniques (Second Edition).*- : Morgan Kaufmann.

Jaafar, Y.; Namly, D.; Bouzoubaa, K.; Yousfi, A. 2017. Enhancing Arabic stemming process using resources and benchmarking tools. *Journal of King Saud University–Computer and Information Sciences 29.* 164-170.

Kodinariya, T.M. dan Makwana, P.R. 2013. Review on Determining Number of Cluster in *K-means* Clustering.*International Journal of Advance Research in Computer Science and Management Studies, 1(6)*

Kusnawi. 2007. *Pengantar Solusi Data Mining.* Seminar Nasional Teknologi, Yogyakarta

Mahmud, M.R.; Afrin, M.; Razzaque, M.A. Miller, E.; Iwashige, J. 2014. *A Rule Based Bengali Stemmer.*International Conference on Advances in Computing,Communications and Informatics. Bangladesh

Meitei, S.P.; Purkayastha, B.S.; Devi, H.M. 2015. *Development of A Manipuri Stemmer: A Hybrid Approach.*International Symposium on Advanced Computing and Communication. Silchar

Mupemhi, S.; Mupemhi, R.; dan Feremba, J.T. 2006. *Understanding Service Failure Management in Marketing Oriented Organisations: A Review.* Department of Marketing at Midlends State University. 2(2), 98-104

Naraadhipa, A.R. dan Purwarianti, A. 2011. *Sentiment Classification for Indonesian Message in Social Media.* 2011 International Conference on Electrical Engineering and Informatics, Bandung.

Nazief, B. 1996. *Confix Stripping: Approach to Stemming Algorithm forBahasa Indonesia*. Internal Publication. Faculty of Computer Science,University of Indonesia, Depok, Jakarta.

Oussalah, M.B.; Escallier; dan Daher, D. 2016. An Automated System for Grammatical Analysis of Twitter Messages: A Learning Task Application.*Knowledge-Based Systems 101,* 31-47.

Paice, C.D. 1996. *An Evaluation Method for Stemming Algorithms,* Department of Computing, Lancaster University.

Pei-wul, D. dan Yan-qiu, H. 2006.*Research of Customer Complaints and Service Recovery Effects.* School of Management, Beijing Institute of Technology, China.

Perikos, I. dan Hatzilygeroudis, I. 2016.Recognizing Emotions in Text Using Ensemble of Classifiers.*Engineering Applications of Artificial Intelligence 51*, 191-201.

Petz, G.; Karpowicz, M.; Fürschuß, H.; Auinger, A.; Strítesky, V.; dan Holzinger, A. 2014. Computational Approaches for Mining User's Opinions on the Web 2.0.*Information Processing and Management 50*, 899–908

Porter, M.F. 1980. *An Algorithm for Suffixes Stripping,* Computer Laboratory, Corn Exchange Street, Cambridge.

Prasetyo, E. dan Timoteus, T. 2007.*Teknik Pemenggalan Kata Bahasa Indonesia dalam Dokumen.*Universitas Gunadarma Jakarta.

Rajput, B.S. dan Khare, N. 2015. A Survey of Stemming Algorithms for Information Retrieval. *IOSR Journal of Computer Engineering 17(3)*, 76-80.

Rendón, E.; Abundez, I.; Arizmendi, A.; dan Quiroz, E.M. 2011.Internal versus External Cluster Validation Indexes.*International Journal of Computers and Communication 5(1)*, 27-34

Sari, Y.A.; Kamilah, E.; Mutrofin, S.; dan Arifin, A.Z. 2014, User Emotion Indentification in Twitter Using Specific Features Hastag, Emoji, Emoticon, and Adjective Term.*Journal of Computer Science and Information7(1),* 18-23.

Seo, W.; Yoon, J.; Park, H.; Coh, B.Y.; Lee, J.M.; dan Kwon, O.J. 2016. Product Opportunity Identification Based on Internal Capabilities Using Text Mining and Association Rule Mining.*Technological Forecasting & Social Change 105,* 94–104.

Shrestha, I.; Dhakal, S.S. 2016. *A New Stemmer For Nepali Language.* 2016 2nd International Conference on Advances in Computing, Communication, & Automation. India.

Shri, T.K.P. dan Sriraam, N. 2017.Comparison of t-test Ranking with PCA and SEPCOR Feature Selection for Wake and Stage 1 Sleep Pattern Recognition in Multichannel Electroencephalograms.*Biomedical Signal Processing and Control 31,* 499-512.

Tala, F.Z. 2003.*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.* Master of Logic Project,Institute for Logic, Language and Computation,Universiteit van Amsterdam.

Tarigan, I.A. 2013. Pengguna Twitter Indonesia Teraktif Ketiga di Dunia.*http://chip.co.id/news/appssocial_media/9030.* Diambil 12 Oktober 2016.

Trappey, C.; Wu, H.Y.; dan Liu, K.L. 2012.*Knowledge Discovery of Customer Satisfaction and Dissatisfaction Using Ontology-based Text Analysis of Critical Incident Dialogues,* Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design.

Trappey, C.V.; Wu, H.Y.; Taghaboni-Dutta, F.; dan Trappey, A.J.C. 2011.Using Patent Data for Technology Forecasting: China RFID Patent Analysis. *Advanced Engineering Informatics 25*, 53–64.

Twitter @pln_123

Urbain, J. 2015.Mining Heart Disease Risk Factors in Clinical Text with Named Entity Recognition and Distributional Semantic Models.*Journal of Biomedical Informatics 58,* 143-149.

Wang, H.; Wu, J.; Yuan, S.; dan Chen, J. 2016. On Characterizing Scale Effect of Chinese Mutual Funds via Text Mining.*Signal Processing 124,* 266–278.

Wasista, S. dan Astin, N. 2010. *Algoritma Sistem Pembaca Teks Bahasa Indonesia MenggunakanMetode FSA (Finite State Automata),* Jurusan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya.

www.pln.co.id

http://hikaruyuuki.lecture.ub.ac.id/kamus-kata-dasar-dan-stopword-list-bahasa-indonesia/

Halaman ini sengaja dikosongkan

Lampiran 1
Kode Program Metode Porter *Confix Stripping Stemmer*

Method cekVowel ( )

```
private boolean cekVowel(String kata){
boolean vowel=false;
char []v={'a','i','u','e','o'};
for(int i=0;i<kata.length();i++){
for(int j=0;j<v.length;j++){
if(kata.charAt(i)==v[j]){
vowel=true;
}}}
return vowel;}
```

Method *cekVowel* digunakan untuk mengecek apakah kata memiliki karakter-karakter vokal. Cara pengecekkan dengan menggunakan dua perulangan. Perulangan pertama untuk mengecek seluruh karakter yang ada pada kata, sedangkan perulangan kedua untuk mencocokkan huruf pada kata dengan huruf vokal pada *array v.*

Method isVowel ( )

```
private boolean isVowel(char huruf){
boolean vowel=false;
char []v={'a','i','u','e','o'};
for(int i=0;i<v.length;i++){
if(huruf==(v[i])){
vowel=true;}}
return vowel;}
```

Method *isVowel* hampir sama dengan method *cekVowel*. Namun pada method ini, masukan hanya berupa satu huruf dan bukan satu kata.

Method isConsonant ( )

```
private boolean isConsonant(char huruf){
boolean consonant=false;
char [] C = {'b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w','x','y','z'};
for(int i=0;i<C.length;i++){
if(huruf==C[i]){
consonant = true;}}
return consonant;}
```

Method *isConsonant* berfungsi sama dengan method *isVowel*, namun khusus untuk mengecek huruf konsonan.

Method cekDouble ( )

```
private boolean cekDouble(String kata){
boolean dkonsonan=false;
if(kata.charAt(kata.length()-1)==kata.charAt(kata.length()-2)){
dkonsonan=true;}
return dkonsonan;}
```

Method *cekDouble* digunakan untuk mengecek apakah pasangan huruf yang dimasukkan merupakan pasangan huruf yang sama.

Method cekCVC ( )

```
private boolean cekCVC(String kata){
PorterStemmer cek=new PorterStemmer();
boolean cvc=false;
if(cek.isConsonant(kata.charAt(kata.length()-
1))==true&&cek.isVowel(kata.charAt(kata.length()-
2))==true&&cek.isConsonant(kata.charAt(kata.length()-3))==true){
cvc=true;}
return cvc;}
```

Method *cekCVC* digunakan untuk mengecek apakah urutan dari huruf yang dimasukkan memenuhi urutan konsonan-vokal-konsonan.

Method cekMeasure ( )

```
private int cekMeasure(String kata){
PorterStemmer cek=new PorterStemmer();
int m=0;
if(kata.length()>1){
for(int i=0;i<kata.length();i++){
if(i==kata.length()-1){
break;}
else
if(cek.isVowel(kata.charAt(i))==true&&cek.isConsonant(kata.charAt(i+1))==t
rue){
m=m+1;
}}}
return m;}
```

Method *cekMeasure* digunakan untuk mengecek ada berapa banyak urutan vokal-konsonan yang ada pada kata. Pada method ini, variabel *integer m* digunakan untuk menampung hasil pengecekkan vokal-konsonan. Apabila panjang kata lebih dari 2 huruf, method ini dijalankan, tetapi jika tidak, maka nilai *m=0.* Pada proses pengecekkan terdapat pengecekan jumlah perulangan agar tidak terjadi *error*, yaitu pada *if(i==kata.length()-1){break}*. Method ini merupakan fungsi yang sangat penting untuk metode *porter stemmer*.

Method step1a ( )

```
private String step1a (String kata){
if(kata.endsWith("sses")){
kata=kata.substring(0, kata.length()-2);}
else if(kata.endsWith("ies")){
kata=kata.substring(0, kata.length()-2);}
else if(kata.endsWith("ss")){
}
else if(kata.endsWith("s")){
kata=kata.substring(0, kata.length()-1);}
else{return kata;}
```

Method *step1a* merupakan salah satu kelompok aturan yang dijalankan pada metode *stemmer porter.* Selain *step1a*, terdapat 7 method lagi yaitu : *step1b, step1c, step2, step3, step4, step5a,* dan *step5b.* Seluruh method tersebut dijalankan berurutan. Berikut adalah kode program untuk method-method tersebut.

Method step1b ( )

```
private String step1b(String kata){
int i=0;
String temp=kata;
PorterStemmer cek=new PorterStemmer();
if((cek.cekMeasure(kata)>0)&&kata.endsWith("eed")){
kata=kata.substring(0, kata.length()-1);}
else if(cek.cekVowel(kata)==true&&kata.endsWith("ed")){
kata=kata.substring(0, kata.length()-2);
i=i+1;}
else if(kata.endsWith("ing")){
kata=kata.substring(0, kata.length()-3);
i=i+1;}
if(i>0&&kata.endsWith("at")){
kata=kata.concat("e");}
else if(i>0&&kata.endsWith("bl")){
kata=kata.concat("e");}
else if(i>0&&kata.endsWith("iz")){
kata=kata.concat("e");}
else
if(i>0&&cek.cekDoubleConsonant(kata)==true&&!kata.endsWith("l")
|!kata.endsWith("s")|!kata.endsWith("z")){
kata=kata.substring(0, kata.length()-1);}
else if(cek.cekMeasure(kata)==1&&cek.cekCVC(kata)==true){
kata=kata.concat("e");}
else{
return kata;}
return kata;}
```

Method step1c ( )

```
private String step1c(String kata){
PorterStemmer stem=new PorterStemmer();
if(stem.cekVowel(kata)==true&&kata.endsWith("y")){
kata=kata.substring(0, kata.length()-1);
kata=kata+'i';}
return kata;}
```

Method step2 ( )

```
private String step2(String kata){
PorterStemmer cek=new PorterStemmer();
if(cek.cekMeasure(kata)>0&&kata.endsWith("ational")){
kata=kata.substring(0, kata.length()-5);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("tional")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("anci")){
kata=kata.substring(0, kata.length()-1);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("izer")){
kata=kata.substring(0, kata.length()-1);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("abli")){
kata=kata.substring(0, kata.length()-1);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("alli")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("entli")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("eli")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ousli")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ization")){
kata=kata.substring(0, kata.length()-5);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ation")){
kata=kata.substring(0, kata.length()-3);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ator")){
kata=kata.substring(0, kata.length()-2);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("alism")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("iveness")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("fulness")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ousness")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("aliti")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("iviti")){
kata=kata.substring(0, kata.length()-3);
kata=kata+'e';}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("biliti")){
kata=kata.substring(0, kata.length()-5);
kata=kata+"le";}
else{
return kata;}
return kata;}
```

Method step3 ( )

```
private String step3(String kata){
PorterStemmer cek=new PorterStemmer();
if(cek.cekMeasure(kata)>0&&kata.endsWith("icate")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ative")){
kata=kata.substring(0, kata.length()-5);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("alize")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("iciti")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ical")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ful")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>0&&kata.endsWith("ness")){
kata=kata.substring(0, kata.length()-4);}
else{
return kata;}
return kata;}
```

Method step4 ( )

```
private String step4(String kata){
PorterStemmer cek=new PorterStemmer();
if(cek.cekMeasure(kata)>1&&kata.endsWith("al")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ance")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ence")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("er")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ic")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("able")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ible")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ant")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ement")){
kata=kata.substring(0, kata.length()-5);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ment")){
kata=kata.substring(0, kata.length()-4);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ent")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ion")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ou")){
kata=kata.substring(0, kata.length()-2);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ism")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ate")){
kata=kata.substring(0, kata.length()-3);}
```

Method step4 ( ) Lanjutan

```
else if(cek.cekMeasure(kata)>1&&kata.endsWith("iti")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ous")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ive")){
kata=kata.substring(0, kata.length()-3);}
else if(cek.cekMeasure(kata)>1&&kata.endsWith("ize")){
kata=kata.substring(0, kata.length()-3);}
else{
return kata;}
```

Method step5a ( )

```
private String step5a(String kata){
PorterStemmer cek=new PorterStemmer();
if(cek.cekMeasure(kata)>1&&kata.endsWith("e")){
kata=kata.substring(0, kata.length()-1);}
else
if(cek.cekMeasure(kata)>1&&cek.cekCVC(kata)&&kata.endsWith("e")
){
kata=kata.substring(0, kata.length()-1);}
else{
return kata;}
```

Method step5b ( )

```
private String step5b(String kata){
PorterStemmer cek=new PorterStemmer();
if(cek.cekMeasure(kata)>1&&cek.cekDoubleConsonant(kata)==true&&
kata.charAt(kata.length())=='l'){
kata=kata.substring(0, kata.length()-1);}
else{
return kata;}
return kata;}
```

Lampiran 2
Kode Program Metode Stemmer Nazief

Pada bagian lampiran ini akan dijelaskan method-method yang ada pada program metode stemmer Nazief.

Method HapusInflectionalParticle ( )

```
public String HapusInflectionalParticle(String kata){
if(kata.endsWith("lah")){
kata=kata.substring(0, kata.length()-3);}
else if(kata.endsWith("kah")){
kata=kata.substring(0, kata.length()-3);}
else if(kata.endsWith("tah")){
kata=kata.substring(0, kata.length()-3);}
else if(kata.endsWith("pun")){
kata=kata.substring(0, kata.length()-3);}
return kata;}
```

Method *HapusInflectionalParticle* digunakan untuk menghapus prefiks *lah, kah, tah,* dan *pun*. Pada method ini, parameter yang menjadi masukan adalah sebuah String kata misalnya "membantah". Kata tersebut akan dilakukan pengecekan aturan dan akan cocok dengan prefik "tah". Kemudian kata tersebut dipotong akhirannya menggunakan kode *substring*, sehingga hasil pemenggalan menjadi "memban".

Method HapusPossesivePronoun ( )

```
public String HapusPossesivePronoun(String kata){
if(kata.endsWith("ku")){
kata=kata.substring(0, kata.length()-2);}
else if(kata.endsWith("mu")){
kata=kata.substring(0, kata.length()-2);}
else if(kata.endsWith("nya")){
kata=kata.substring(0, kata.length()-3);}
return kata;}
```

Method *HapusPossesivePronoun* digunakan untuk menghapus prefiks *ku, mu,* dan *nya.* Proses yang terjadi pada method ini sama dengan method HapusInflectionalParticle.

Method HapusDerivationalSuffix ( )

```
public String HapusDerivationalSuffix(String kata){
if(kata.endsWith("i")){
kata=kata.substring(0, kata.length()-1);}
else if(kata.endsWith("kan")){
kata=kata.substring(0,kata.length()-3);}
else if(kata.endsWith("an")){
kata=kata.substring(0,kata.length()-2);}
return kata;}
```

Method *HapusDerivationSuffix* digunakan untuk menghapus imbuhan *i, kan,* dan *an.* Misal inputan adalah "membatalkan" maka *return* dari method tersebut adalah "membatal".

Method cekVokal ( )

```
private boolean cekVokal (char huruf){
boolean vokal = false;
char [] V = {'a','i','u','e','o'};
for(int i=0;i<V.length;i++){
if(huruf==V[i]){
vokal = true;}
}
return vokal;}
```

Method *cekVokal* digunakan untuk mengecek apakah karakter yang dijadikan parameter termasuk karakter huruf vokal. Untuk proses pengecekan digunakan perulangan yang akan mencocokkan karakter inputan dengan *array V* yang berisi huruf-huruf vokal. Hasil dari method ini adalah nilai *true* jika karakter adalah vokal, dan nilai *false* jika bukan vokal.

Method cekKonsonan ( )

```
private boolean cekKonsonan (char huruf){
boolean konsonan = false;
char [] C =
{'b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t
','v','w','x','y','z'};
for(int i=0;i<C.length;i++){
if(huruf==C[i]){
konsonan = true;}
}
return konsonan;}
```

Sama seperti method *cekVokal*, method *cekKonsonan* akan mencocokkan karakter inputan dengan *array C* yang berisi huruf-huruf konsonan. Method ini juga akan memberikan nilai *true* jika karakter inputan adalah konsonan.

Method cekSisipan ( )

```
private boolean cekSisipan (String kata){
boolean sisipan = false;
String P[]={"in","el","em","er","ah"};
for(int i=0;i<P.length;i++){
if(kata.equals(P[i])){
sisipan = true;}
}
return sisipan;}
```

Method cekSisipan digunakan untuk mengecek apakah String yang dimasukkan adalah sisipan. Terdapat lima sisipan yang didefinisikan pada mothod ini yaitu *in, el, em, er,* dan *ah.* Keluaran dari method ini adalah nilai *true* atau *false.*

Method HapusPrefiksNazief ( )

```
public String HapusPrefiksNazief (String kata){
 String temp=kata;
 DBConnect con = new DBConnect();
try{
if(kata.startsWith("di")){
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("ke")){
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("se")){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else if
(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==true&
&kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true){
kata=kata.substring(3, kata.length());}
else if(kata.equals("belajar")){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("be")&&con.cekKonsonan(kata.charAt(2))==true
&&kata.charAt(2)!='r'&&kata.charAt(2)!='l'&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("ter")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("ter")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(4,
6).equals("er")&&con.cekVokal(kata.charAt(6))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("ter")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&!kata.substring(4, 6).equals("er")){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("te")&&con.cekKonsonan(kata.charAt(2))==true
&&kata.charAt(2)!='r'&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("mempe")&&kata.charAt(5)=='r'|kata.charAt(5)
=='l'){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("memp")&&con.cekVokal(kata.charAt(4))==true&
&kata.charAt(4)!='e'){
kata=kata.substring(3, kata.length());}
```

Method HapusPrefiksNazief ( ) Lanjutan

```
else
if(kata.startsWith("meny")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());
kata='s'+kata;}
else
if(kata.startsWith("meng")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());}
else
if(kata.startsWith("meng")&&con.ghq(kata.charAt(4))==true){
kata=kata.substring(4, kata.length());}
else
if(kata.startsWith("men")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("men")&&con.cdjz(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("mem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("mem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(3))==true){
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("mem")&&con.bfv(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("me")&&con.lrwy(kata.charAt(2))==true&&con.c
ekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("pelajar")){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("peng")&&con.ghq(kata.charAt(4))==true){
kata=kata.substring(4, kata.length());}
else
if(kata.startsWith("peng")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());}
else
if(kata.startsWith("peny")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());
kata='s'+kata;}
else
if(kata.startsWith("per")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("per")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("per")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true&&kata.charA
t(3)!='r'){
kata=kata.substring(3, kata.length());}
```

Method HapusPrefiksNazief ( ) Lanjutan

```
else if(kata.startsWith("pem")&&con.bfv(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("pem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("pem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(4))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("pen")&&con.cdjz(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("pen")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("pel")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("pe")&&kata.charAt(2)=='w'&&con.cekVokal(kat
a.charAt(3))==true||kata.startsWith("pe")&&kata.charAt(2)=='y'&
&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("pe")&&con.rwylmn(kata.charAt(2))==true&&kat
a.substring(3,
5).equals("er")&&con.cekVokal(kata.charAt(5))==true){
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("pe")&&con.rwylmn(kata.charAt(2))==true&&!ka
ta.substring(3, 5).equals("er")){
kata=kata.substring(2, kata.length());}
}catch(Exception e){
return temp;}
return kata;}
```

Method *HapusPrefiksNazief* digunakan untuk menghapus suffiks atau awalan dari kata. Inputan method ini adalah sebuah String kata yang kemudian dilakukan pengecekan aturan penghapusan awalan sesuai aturan pada Tabel 2.1 yang berjumlah 33 aturan, ditambah penghapusan awalan *di*, *ke*, dan *se* yang dapat dihapus langsung. Apabila kata yang dimasukkan sesuai dengan aturan yang telah didefinisikan, maka akan dilakukan pemotongan awalan sesuai panduan pada Tabel 2.1. Apabila parameter yang dimasukkan tidak sesuai dengan seluruh aturan yang ada, maka keluaran method sama dengan masukan kata awal.

Method PrefiksRecoding ( )

```
public String PrefiksRecoding (String kata){
DBConnect con = new DBConnect();
String temp=kata;
try{
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("ter")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("mem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("mem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(3))==true){
kata=kata.substring(3, kata.length());
kata='p'+kata;}
else
if(kata.startsWith("men")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());
kata='t'+kata;}
else
if(kata.startsWith("meng")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());
kata='k'+kata;}
else
if(kata.startsWith("per")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("pem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("pem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(4))==true){
kata=kata.substring(3, kata.length());
kata='p'+kata;}
else
if(kata.startsWith("pen")&&con.cekVokal(kata.charAt(3))==true){
kata=kata.substring(3, kata.length());
kata='t'+kata;}
else
if(kata.startsWith("peng")&&con.cekVokal(kata.charAt(4))==true)
{
kata=kata.substring(4, kata.length());
kata='k'+kata;}
}catch(Exception e){
return temp;}
return kata;}
```

Method *PrefiksRecoding* dijalankan apabila kata belum ditemukan pada kamus kata dasar setelah penghapusan prefiks. Kata akan dikembalikan menjadi sebelum penghapusan prefiks, kemudian dicoba menghapus prefiks dengan aturan *recoding*. Pada aturan recoding terdapat 9 aturan yang berbeda dengan penghapusan prefiks biasa, dan diantaranya ada 7 aturan dimana awal kata dasar diganti dengan karakter recoding. Contoh : *pengabut*cocok dengan aturan `if(kata.startsWith("peng")&&con.cekVokal(kata.charAt(4))==true)` maka *pengabut* menjadi *abut* kemudian menjadi *kabut.*

Method cekPreffixNazief ( )

```
public String cekPreffixNazief (String kata){
 DBConnect con = new DBConnect();
String preffix="";
try{
if(kata.startsWith("di")){
preffix="di";}
else if(kata.startsWith("ke")){
preffix="ke";}
else if(kata.startsWith("se")){
preffix="se";}
else
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){
//berV
preffix="ber";}
else if
(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==true&
&kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){
preffix="ber";}
else
if(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true){
preffix="ber";}
else if(kata.startsWith("belajar")){
preffix="bel";}
else
if(kata.startsWith("be")&&con.cekKonsonan(kata.charAt(2))==true
&&kata.charAt(2)!='r'&&kata.charAt(2)!='l'&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){
preffix="be";}
else
if(kata.startsWith("ter")&&con.cekVokal(kata.charAt(3))==true){
preffix="ter";}
else
if(kata.startsWith("ter")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(4,
6).equals("er")&&con.cekVokal(kata.charAt(6))==true){
preffix="ter";}
else
if(kata.startsWith("ter")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&!kata.substring(4, 6).equals("er")){
preffix="ter";}
else
if(kata.startsWith("te")&&con.cekKonsonan(kata.charAt(2))==true
&&kata.charAt(2)!='r'&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){
preffix="te";}
else
if(kata.startsWith("mempe")&&kata.charAt(5)=='r'|kata.charAt(5)
=='l'){
preffix="mem";}
else
if(kata.startsWith("memp")&&con.cekVokal(kata.charAt(4))==true&
&kata.charAt(4)!='e'){
preffix="mem";}
```

Method cekPreffixNazief ( ) Lanjutan

```
else
if(kata.startsWith("meny")&&con.cekVokal(kata.charAt(4))==true)
{
preffix="meny";}
else
if(kata.startsWith("meng")&&con.cekVokal(kata.charAt(4))==true)
{
preffix="meng";}
else
if(kata.startsWith("meng")&&con.ghq(kata.charAt(4))==true){
preffix="meng";}
else
if(kata.startsWith("men")&&con.cekVokal(kata.charAt(3))==true){
preffix="me";}
else
if(kata.startsWith("men")&&con.cdjz(kata.charAt(3))==true){
preffix="men";}
else
if(kata.startsWith("mem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("mem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(3))==true){
preffix="me";}
else if(kata.startsWith("mem")&&con.bfv(kata.charAt(3))==true){
preffix="mem";}
else
if(kata.startsWith("me")&&con.lrwy(kata.charAt(2))==true&&con.c
ekVokal(kata.charAt(3))==true){
preffix="me";}
else if(kata.startsWith("pelajar")){
preffix="pel";}
else
if(kata.startsWith("peng")&&con.ghq(kata.charAt(4))==true){
preffix="peng";}
else
if(kata.startsWith("peng")&&con.cekVokal(kata.charAt(4))==true)
{
preffix="peng";}
else
if(kata.startsWith("peny")&&con.cekVokal(kata.charAt(4))==true)
{
preffix="peny";}
else
if(kata.startsWith("per")&&con.cekVokal(kata.charAt(3))==true){
preffix="per";}
else
if(kata.startsWith("per")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){
preffix="per";}
else
if(kata.startsWith("per")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true&&kata.charA
t(3)!='r'){
preffix="per";}
else if(kata.startsWith("pem")&&con.bfv(kata.charAt(3))==true){
```

Method cekPreffixNazief ( ) Lanjutan

```
else
if(kata.startsWith("pen")&&con.cdjz(kata.charAt(3))==true){
preffix="pen";}
else
if(kata.startsWith("pen")&&con.cekVokal(kata.charAt(3))==true){
preffix="pe";}
else
if(kata.startsWith("pel")&&con.cekVokal(kata.charAt(3))==true){
preffix="pe";}
else
if(kata.startsWith("pe")&&kata.charAt(2)=='w'&&con.cekVokal(kat
a.charAt(3))==true||kata.startsWith("pe")&&kata.charAt(2)=='y'&
&con.cekVokal(kata.charAt(3))==true){
preffix="pe";}
else
if(kata.startsWith("pe")&&con.rwylmn(kata.charAt(2))==true&&kat
a.substring(3,
5).equals("er")&&con.cekVokal(kata.charAt(5))==true){
preffix="per";}
else
if(kata.startsWith("pe")&&con.rwylmn(kata.charAt(2))==true&&!ka
ta.substring(3, 5).equals("er")){
preffix="pe";}
}catch(Exception e){
return preffix;}
return preffix;}
```

Method *cekPreffixNazief*digunakan untuk mendeteksi prefik apa yang ada pada kata. Method ini dibutuhkan untuk proses perulangan penghapusan prefiks dimana ada kondisi apabila prefiks yang sudah dihapus sama dengan prefiks yang akan dihapus, maka proses dihentikan.

Lampiran 3
Kode Program Metode *Arifin Stemmer*

Metode *arifin stemmer* merupakan metode *stemmer* yang hampir sama dengan metode *nazief*. Perbedaan utama adalah urutan penghapusan imbuhan dan adanya proses kombinasi prefiks dan sufiks. Metode ini juga menggunakan kamus kata dasar sebagai kontrol *understemming*, *overstemming*. Berikut kode program untuk method-method yang ada pada metode *arifin stemmer*.

Method PotongAwalan ( )

```
public String PotongAwalan(String kata){
DBConnect con = new DBConnect();
String temp=kata;
try{
if(kata.startsWith("di")){//1
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("ke")){//2
kata=kata.substring(2, kata.length());}
else if(kata.startsWith("se")){//3
kata=kata.substring(2, kata.length());}
else
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){/
/b4
kata=kata.substring(3, kata.length());}
else if
(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==true&&
kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){//5
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==true
&&kata.charAt(3)!='r'&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true){//6
kata=kata.substring(3, kata.length());}
else if(kata.startsWith("belajar")){//7
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("be")&&con.cekKonsonan(kata.charAt(2))==true&
&kata.charAt(2)!='r'&&kata.charAt(2)!='l'&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){//8
kata=kata.substring(2, kata.length());}
```

Method *PotongAwalan* digunakan untuk menghapus prefiks pada metode *arifin*. Aturan pemenggalan yang digunakan sama seperti metode *nazief* sehingga menggunakan kode program yang sama.

Method PotongAkhiran ( )

```
public String PotongAkhiran(String kata){
String temp=kata;
 try{
 if(kata.endsWith("lah")){
kata=kata.substring(0, kata.length()-3);}
if(kata.endsWith("kah")){
kata=kata.substring(0, kata.length()-3);}
if(kata.endsWith("tah")){
kata=kata.substring(0, kata.length()-3);}
if(kata.endsWith("pun")){
```

Method PotongAkhiran ( ) Lanjutan

```
kata=kata.substring(0, kata.length()-3);}
if(kata.endsWith("ku")){
kata=kata.substring(0, kata.length()-2);}
if(kata.endsWith("mu")){
kata=kata.substring(0, kata.length()-2);}
if(kata.endsWith("nya")){
kata=kata.substring(0, kata.length()-3);}
if(kata.endsWith("i")){
kata=kata.substring(0, kata.length()-1);}
if(kata.endsWith("kan")){
kata=kata.substring(0,kata.length()-3);}
else if(kata.endsWith("an")){
kata=kata.substring(0,kata.length()-2);}
}catch(Exception e){
return temp;}
return kata;}
```

Method *PotongAkhiran* digunakan untuk menghapus sufiks dari kata. Pada metode *arifin*, aturan penghapusan akhiran atau sufiks sama dengan aturan penghapusan pada metode *nazief stemmer*. Perbedaanya pada metode *arifin*, seluruh aturan penghapusan sufiks dijadikan satu method.

Method getPrefiks ( )

```
public String getPrefiks(String kata){
String prefiks="";
DBConnect con = new DBConnect();
try{
if(kata.startsWith("di")){//1
prefiks="di";}
else if(kata.startsWith("ke")){//2
prefiks="ke";}
else if(kata.startsWith("se")){//3
prefiks="se";}
else
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){
//b4
prefiks="ber";}
else if
(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==true&
&kata.charAt(3)!='r'&&!kata.substring(5, 7).equals("er")){//5
prefiks="ber";}
else
if(kata.startsWith("ber")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(5,
7).equals("er")&&con.cekVokal(kata.charAt(7))==true){//6
prefiks="ber";}
```

Method *getPrefiks* digunakan untuk menyalin prefiks yang ada pada kata sebelum prefiks tersebut dihapus dengan method *PotongAwalan*. Prefiks yang didapatkan tersebut disimpan dalam variabel *prefik1* dan *prefik2*. Prefiks tersebut nantinya akan digunakan pada tahap kombinasi prefiks-sufiks. Aturan yang

digunakan pada method *getPrefiks* sama dengan aturan yang digunakan pada method *PotongAwalan.*

Method getSufiks ( )

```
public String getSufiks(String kata){//stemer arifin
String sufiks="";
try{
if(kata.endsWith("lah")){
sufiks="lah";}
if(kata.endsWith("kah")){
sufiks="kah";}
if(kata.endsWith("tah")){
sufiks="tah";}
if(kata.endsWith("pun")){
sufiks="pun";}
if(kata.endsWith("ku")){
sufiks="ku";}
if(kata.endsWith("mu")){
sufiks="mu";}
if(kata.endsWith("nya")){
sufiks="nya";}
if(kata.endsWith("i")){
sufiks="i";}
if(kata.endsWith("kan")){
sufiks="kan";}
else if(kata.endsWith("an")){
sufiks="an";}
}catch(Exception e){
return sufiks;}
return sufiks;}
```

Method *getSufiks* berfungsi untuk menyalin akhiran dari kata. Aturan yang digunakan sama dengan method *PotongAkhiran.* Sufiks yang disalin akan disimpan pada variabel *sufik1, sufik2,* dan *sufik3.*

Method getPrefiksRecoding ( )

```
public String getPrefiksRecoding(String kata){//stemer arifin
String recoding="";
DBConnect con = new DBConnect();
try{
if(kata.startsWith("ber")&&con.cekVokal(kata.charAt(3))==true){
recoding="be";}
else
if(kata.startsWith("ter")&&con.cekVokal(kata.charAt(3))==true){
recoding="te";}
else
if(kata.startsWith("mem")&&con.cekVokal(kata.charAt(3))==true||
kata.startsWith("mem")&&kata.charAt(3)=='r'&&con.cekVokal(kata.
charAt(3))==true){
recoding="me";}
else
if(kata.startsWith("men")&&con.cekVokal(kata.charAt(3))==true){
recoding="me";}
else
```

Method *getPrefiksRecoding* berfungsi sama seperti method *getPrefiks.* Method ini digunakan saat melakukan recoding pada metode *arifin.*

Kode Program Kombinasi Awalan-Akhiran Metode *Arifin Stemmer*

```
KD=prefik1+prefik2+kata;
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
else{//kombinasi2
KD="";
KD=prefik1+prefik2+kata+sufik3;
System.out.println("kata 2 : "+KD);
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
else{
KD="";
KD=prefik1+prefik2+kata+sufik3+sufik2;
System.out.println("kata 3 : "+KD);
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
else{
KD="";
KD=prefik2+kata;
System.out.println("kata 4 : "+KD);
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
else{
KD="";
KD=prefik2+kata+sufik3;
System.out.println("kata 5 : "+KD);
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
else{
KD="";
KD=prefik2+kata+sufik3+sufik2;
System.out.println("kata 6 : "+KD);
if(con.cekKataDasar(con.connect(), KD)==true){
katadasar=KD;
stemstop=true;}
```

Kode program tersebut digunakan untuk melakukan tahap kombinasi awalan-akhiran dimana sufiks dan prefiks yang telah dihapus dipasangkan kembali pada kata dasar. Proses ini dilakukan dengan tujuan mengecek apakah ada kata dasar yang terhapus sebagian saat tahap penghapusan imbuhan dilakukan. Pada tahap ini, imbuhan yang tersimpan pada variabel *sufik1*, *sufik2*, *prefik1*, *prefik2*, dan *prefik3* dicoba dikembalikan pada kata dasar. Namun tahap ini hanya dijalankan apabila tahap penghapusan imbuhan gagal.

Lampiran 4
Kode Program Metode *Fadillah Stemmer*

Metode *fadillah stemmer* merupakan jenis metode *confix striping* yang mirip dengan metode *porter*. Hal tersebut karena metode *fadillah* tidak menggunakan kamus kata dasar seperti metode *confix striping stemmer* bahasa Indonesia yang telah ada sebelumnya. Perbedaan antara metode *porter* dan *fadillah* adalah penggunaan suku kata sebagai kontrol proses *stemming*, sedangkan metode *porter* menggunakan jumlah pasangan vokal-konsonan yang disebut *measure (m)*. Berikut ini adalah method-method program yang digunakan pada metode *fadillah stemmer*.

Method cekVokal ( )

```
private boolean cekVokal(char c) {
boolean vokal = false;
char[] v = {'a', 'i', 'u', 'e', 'o'};
for (int i = 0; i < v.length; i++) {
if (c == v[i]) {
vokal = true;}}
return vokal;}
```

Method *cekVokal* digunakan untuk mengecek karakter hutuf yang dimasukkan pada parameter apakah merupakan huruf vokal. Method ini menghasilkan nilai *true* jika masukan adalah huruf vokal.

Method pecahKata ( )

```
public String[][] pecahKata(String s) {
String[][] suku = null;
suku = new String[20][2];
int p = 0;
for (int i = 0; i < s.length(); i++) {
if (cekVokal(s.charAt(i)) == true) {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "v";
p++;
} else {
if (s.charAt(i) == 'n') {
if (i == s.length() - 1) {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;
} else if (cekVokal(s.charAt(i + 1)) == true) {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "kv";
p++;
i++;
} else if (s.charAt(i + 1) == 'g' || s.charAt(i + 1) == 'y') {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "k";
p++;
i++;
```

Method pecahKata ( ) Lanjutan

```
} else {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;}}
else if (s.charAt(i) == 'k') {
if (i == s.length() - 1) {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;
} else if (cekVokal(s.charAt(i + 1)) == true) {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "kv";
p++;
i++;
} else if (s.charAt(i + 1) == 'h') {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "k";
p++;
i++;
} else {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;
}
} else if (s.charAt(i) == 's') {
if (i == s.length() - 1) {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;
} else if (cekVokal(s.charAt(i + 1)) == true) {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "kv";
p++;
i++;
} else if (s.charAt(i + 1) == 'y') {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "k";
p++;
i++;
} else {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;}
} else {
if (i == s.length() - 1) {
suku[p][0] = Character.toString(s.charAt(i));
suku[p][1] = "k";
p++;
} else if (cekVokal(s.charAt(i + 1)) == true) {
suku[p][0] = Character.toString(s.charAt(i)) +
Character.toString(s.charAt(i + 1));
suku[p][1] = "kv";
```

Method pecahKata ( ) Lanjutan

```
public String[][] pecahKata(String[][] s) {
String[][] suku = new String[20][2];
int p = 0;
int length = 0;
for (int i = 0; i < s.length; i++) {
    if (s[i][0] != null) {
length++;}}
for (int i = 0; i < length; i++) {
    if (s[i][1].equalsIgnoreCase("v")) {
if (i == length - 1) {
suku[p][0] = s[i][0];
suku[p][1] = s[i][1];
} else if (s[i + 1][1].equalsIgnoreCase("k")) {
suku[p][0] = s[i][0] + s[i + 1][0];
suku[p][1] = s[i][1]+s[i+1][1];
p++;
i++;}
else{
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}}
    else if (s[i][1].equalsIgnoreCase("k")) {
if (i == length - 1) {
suku[p][0] = s[i][0];
suku[p][1] = s[i][1];}
else if (s[i + 1][1].equalsIgnoreCase("k")) {
int j = i + 1;
String tmpHur = s[i][0];
String tmpVK = s[i][1];
while (j < length&&s[j][1].equalsIgnoreCase("k") ) {
    tmpHur = tmpHur + s[j][0];
    tmpVK = tmpVK + s[j][1];
    i++;
    j++;}
if (i + 1 < length && s[i + 1][1].equalsIgnoreCase("kv")) {
    if (i + 2 < length && s[i + 2][1].equalsIgnoreCase("k") ) {
tmpHur = tmpHur + s[i + 1][0] + s[i+2][0];
tmpVK = tmpVK + s[i + 1][1] + s[i + 2][1];
suku[p][0] = tmpHur;
suku[p][1] = tmpVK;
i = i + 2;
p++;}
    else{
suku[p][0]=tmpHur+s[i+1][0];
suku[p][1]=tmpVK+s[i+1][1];
i++;
p++;}}
else{
    suku[p][0]=tmpHur;
    suku[p][1]=tmpVK;
    p++;}}
else if(i+1<length && s[i+1][1].equalsIgnoreCase("kv")){
if (i + 2 < length && s[i + 2][1].equalsIgnoreCase("k")) {
    suku[p][0] = s[i][0]+s[i + 1][0] + s[i + 2][0];
    suku[p][1] = s[i][1]+s[i + 1][1] + s[i + 2][1];
    i = i + 2;
    p++;}
```

Method pecahKata ( ) Lanjutan

```
else{
suku[p][0]=tmpHur+s[i+1][0];
suku[p][1]=tmpVK+s[i+1][1];
i++;
p++;}}
else{
    suku[p][0]=tmpHur;
    suku[p][1]=tmpVK;
    p++;}}
else if(i+1<length && s[i+1][1].equalsIgnoreCase("kv")){
if (i + 2 < length && s[i + 2][1].equalsIgnoreCase("k")) {
    suku[p][0] = s[i][0]+s[i + 1][0] + s[i + 2][0];
    suku[p][1] = s[i][1]+s[i + 1][1] + s[i + 2][1];
    i = i + 2;
    p++;}
else{
    suku[p][0] = s[i][0]+s[i + 1][0];
    suku[p][1] = s[i][1]+s[i + 1][1];
    i++;
    p++;}}
    } else if (s[i][1].equalsIgnoreCase("kv")) {
if (i + 1 < length && s[i + 1][1].equalsIgnoreCase("k")) {
suku[p][0] = s[i][0] + s[i + 1][0];
suku[p][1] = s[i][1] + s[i + 1][1];
i++;
p++;}
else{
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}}}
return suku;}
```

Method pecahKata ( ) Lanjutan

```
public String[][] pecahKata(String[][] s){
String[][] suku = new String[20][2];
int p=0;
int length=0;
for(int i=0;i<s.length;i++){
    if(s[i][0]!=null){
length++;}}
for(int i=0;i<length;i++){

if(s[i][1].equalsIgnoreCase("vk")||s[i][1].equalsIgnoreCase("kv
k") || s[i][1].equalsIgnoreCase("kkvk")){
if(i+1<length && s[i+1][1].equalsIgnoreCase("k")){
suku[p][0]=s[i][0]+s[i+1][0];
suku[p][1]=s[i][1]+s[i+1][0];
i++;
p++;
}else{
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}}
```

Method pecahKata ( ) Lanjutan

```
else if(s[i][1].equalsIgnoreCase("kv") ||
s[i][1].equalsIgnoreCase("kkv") ||
s[i][1].equalsIgnoreCase("kkkv")){
if(i+1<length && s[i+1][1].equalsIgnoreCase("v")){
suku[p][0]=s[i][0]+s[i+1][0];
suku[p][1]=s[i][1]+s[i+1][1];
i++;
p++;
}else{
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}}
    else if(s[i][1].equalsIgnoreCase("v")){
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}
    else{
suku[p][0]=s[i][0];
suku[p][1]=s[i][1];
p++;}}
return suku;}
```

Method *pecahKata* digunakan untuk memecah kata menjadi suku kata. Pada method ini kata akan dimasukkan ke dalam *array* dua dimensi dimana dimensi kedua digunakan untuk menyimpan label dari satiap huruf yang ada pada kata. Untuk melakukan pemecahan kata menjadi suku kata, pertama dilakukan pelabelan pasangan huruf pada kata dimana simbol *v* digunakan pada vokal dan *k* digunakan untuk melabeli huruf konsonan. Setelah setiap huruf diberikan label, dilakukan proses pemenggalan suku kata dengan menjalankan aturan-aturan pemenggalan suku kata yang telah didefinisikan oleh Sari (2011). Keluaran dari proses ini adalah jumlah suku kata dari kata yang dimasukkan sebagai parameter method.

Method hapuspartikel ( )

```
private  String hapuspartikel(String s){
String stemming=s;
SukuKata sk = new SukuKata();
if (s.substring(s.length() - 3,
s.length()).equalsIgnoreCase("kah")
|| s.substring(s.length() - 3,
s.length()).equalsIgnoreCase("lah")
|| s.substring(s.length() - 3,
s.length()).equalsIgnoreCase("pun")) {
if(sk.pecahKata(s.substring(0,s.length() -3 ))>=2){
stemming = s.substring(0,s.length() - 3);}}
else{
return stemming;}
return stemming;}
```

Method *hapuspartikel* digunakan untuk menghapus sufiks *lah, kah,* dan *pun.* Jika suku kata lebih dari atau sama dengan 2, tahap tersebut dijalankan.

Method hapuspp ( )

```
private  String hapuspp(String s){
String stemming=s;
SukuKata sk = new SukuKata();
if(s.substring(s.length()-2,
s.length()).equalsIgnoreCase("ku")||s.substring(s.length()-2,
s.length()).equalsIgnoreCase("mu")){
if(sk.pecahKata(s.substring(0, s.length()-2))>=2){
stemming = s.substring(0,s.length()-2);}}
else if(s.substring(s.length()-3,
s.length()).equalsIgnoreCase("nya")){
if(sk.pecahKata(s.substring(0, s.length()-3))>=2){
stemming = s.substring(0,s.length()-3);}}
return stemming;}
```

Method *hapuspp* digunakan untuk menghapus sufiks tipe *possesive pronoun,* yaitu *ku*, *mu*, dan *nya.*

Method hapusawalan1 ( )

```
private  String hapusawalan1(String s){
String stemming=s;
SukuKata sk = new SukuKata();
if(s.length()<4){
return stemming;}
if(s.substring(0,4).equalsIgnoreCase("meng")){
if(sk.pecahKata(s.substring(4,s.length()))>=2&&s.substring(4,
5).equalsIgnoreCase("a")||s.substring(4,
5).equalsIgnoreCase("i")||s.substring(4,
5).equalsIgnoreCase("e")||s.substring(4,
5).equalsIgnoreCase("u")||s.substring(4,
5).equalsIgnoreCase("o")){
stemming="k"+s.substring(4,s.length());}
else if(sk.pecahKata(s.substring(4,s.length()))>=2){
stemming=s.substring(4,s.length());}}
else if(s.substring(0, 4).equalsIgnoreCase("meny")){
if(sk.pecahKata(s.substring(4, s.length()))>=2){
stemming = "s"+s.substring(4, s.length());}}
else if (s.substring(0, 3).equalsIgnoreCase("men")){
if(sk.pecahKata(s.substring(3, s.length()))>=2&&s.substring(3,
4).equalsIgnoreCase("a")||s.substring(3,
4).equalsIgnoreCase("i")||s.substring(3,
4).equalsIgnoreCase("e")||s.substring(3,
4).equalsIgnoreCase("u")||s.substring(3,
4).equalsIgnoreCase("o")){
stemming ="t"+s.substring(3,s.length());}
else if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming=s.substring(3,s.length());}}
```

Method hapusawalan1 ( ) Lanjutan

```
else if (s.substring(0,3).equalsIgnoreCase("mem")){
if(s.substring(3, 4).equalsIgnoreCase("a")||s.substring(3,
4).equalsIgnoreCase("i")||s.substring(3,
4).equalsIgnoreCase("e")||s.substring(3,
4).equalsIgnoreCase("u")||s.substring(3,
4).equalsIgnoreCase("o")){
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = "p" + s.substring(3, s.length());}}
else{
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = s.substring(3, s.length());}  }}
else if(s.substring(0, 2).equalsIgnoreCase("me")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2,s.length());}   }
else if(s.substring(0,4).equalsIgnoreCase("peng")){
if(sk.pecahKata(s.substring(4, s.length()))>=2){
stemming = s.substring(4, s.length());}}
else if (s.substring(0,4).equalsIgnoreCase("peny")){
if(s.substring(4, 5).equalsIgnoreCase("a")||s.substring(4,
5).equalsIgnoreCase("i")||s.substring(4,
5).equalsIgnoreCase("e")||s.substring(4,
5).equalsIgnoreCase("u")||s.substring(4,
5).equalsIgnoreCase("o")){
if (sk.pecahKata(s.substring(4, s.length())) >= 2) {
stemming = "s" + s.substring(4, s.length());}}
else{
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = s.substring(3, s.length());}}}
else if (s.substring(0, 3).equalsIgnoreCase("pen")){
if(s.substring(3, 4).equalsIgnoreCase("a")||s.substring(3,
4).equalsIgnoreCase("i")||s.substring(3,
4).equalsIgnoreCase("e")||s.substring(3,
4).equalsIgnoreCase("u")||s.substring(3,
4).equalsIgnoreCase("o")){
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = "t" + s.substring(3, s.length());}}
else{
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = s.substring(3, s.length());}}}
else if (s.substring(0, 3).equalsIgnoreCase("pem")){
if(s.substring(3, 4).equalsIgnoreCase("a")||s.substring(3,
4).equalsIgnoreCase("i")||s.substring(3,
4).equalsIgnoreCase("e")||s.substring(3,
4).equalsIgnoreCase("u")||s.substring(3,
4).equalsIgnoreCase("o")){
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = "p" + s.substring(3, s.length());}}
else{
if (sk.pecahKata(s.substring(3, s.length())) >= 2) {
stemming = s.substring(3, s.length());}}}
else if(s.substring(0, 2).equalsIgnoreCase("di")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2, s.length());}}
```

Method hapusawalan1 ( ) Lanjutan

```
else if(s.substring(0, 3).equalsIgnoreCase("ter")){
if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming = s.substring(3, s.length());}}
else if(s.substring(0, 2).equalsIgnoreCase("ke")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2, s.length());}}
return stemming;}
```

Method *hapusawalan1* digunakan untuk menghapus prefiks kelompok pertama, yaitu *meng, meny, men, mem, me, peng, peny, pen, pem, di, ter,* dan *ke.* Untuk mencegah adanya *error*, sebelum dilakukan penghapusan prefiks, perlu dilakukan pengecekkan panjang kata. Jika kata berjumlah lebih dari 4, maka tahap tersebut akan dilakukan. Hal ini disesuaikan dengan jumlah huruf pada awalan yang terbanyakyaitu4.

Method hapusawalan2 ( )

```
private  String hapusawalan2(String s){
String stemming = s;
SukuKata sk = new SukuKata();
if(s.substring(0, 3).equalsIgnoreCase("ber")){
if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming = s.substring(3, s.length());}}
else if(s.substring(0, 3).equalsIgnoreCase("bel")){
if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming = s.substring(3, s.length());}}
else if(s.substring(0, 2).equalsIgnoreCase("be")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2, s.length());}}
else if(s.substring(0, 3).equalsIgnoreCase("per")){
if(sk.pecahKata(s.substring(3, s.length()))>=2&&s.substring(3,
4).equalsIgnoreCase("a")||s.substring(3,
4).equalsIgnoreCase("i")||s.substring(3,
4).equalsIgnoreCase("e")||s.substring(3,
4).equalsIgnoreCase("u")||s.substring(3,
4).equalsIgnoreCase("o")){
stemming ="r"+s.substring(3, s.length());}
else if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming = s.substring(3,s.length());}}
else if(s.substring(0, 2).equalsIgnoreCase("pe")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2, s.length());}}
else if(s.substring(0, 3).equalsIgnoreCase("pel")){
if(sk.pecahKata(s.substring(3, s.length()))>=2){
stemming = s.substring(3, s.length());}}
else if(s.substring(0, 2).equalsIgnoreCase("se")){
if(sk.pecahKata(s.substring(2, s.length()))>=2){
stemming = s.substring(2, s.length());}}
return stemming;}
```

Method *hapusawalan2* digunakan untuk menghapus prefiks kelompok kedua yaitu *ber, bel, be, per, pe, pel,* dan *se.*

Method hapusakhiran ( )

```
private  String hapusakhiran(String s){
String stemming = s;
SukuKata sk = new SukuKata();
if(s.substring(s.length()-3,
s.length()).equalsIgnoreCase("kan")){
if(sk.pecahKata(s.substring(0, s.length()-3))>=2){
stemming = s.substring(0, s.length()-3);}}
else if(s.substring(s.length()-1,
s.length()).equalsIgnoreCase("i")){
   if(!test.substring(0,
3).equalsIgnoreCase("ber")&&!test.substring(0,
2).equalsIgnoreCase("be")&&!test.substring(0,
4).equalsIgnoreCase("peng")){
if(sk.pecahKata(s.substring(0, s.length()-1))>=2){
stemming = s.substring(0, s.length()-1);}}}
else if(s.substring(s.length()-2,
s.length()).equalsIgnoreCase("an")){
if(sk.pecahKata(s.substring(0, s.length()-2))>=2){
stemming = s.substring(0, s.length()-2);}}
return stemming;}
```

Method *hapusakhiran* digunakan untuk menghapus sufiks *kan*, *i* dan *an*. Khusus untuk akhiran *i* dilakukan pengecekan awalan terlebih dahulu karena akhiran tersebut tidak boleh berpasangan dengan prefiks *ber*, *be,* dan *peng.*

Kode Program Implementasi Metode Fadillah *Stemmer*

```
public String stemmerTala(String s){
   String stem=s;
   try{
   test=s;
if (s.length() >= 3) {
stem = hapusawalan1(hapuspp(hapuspartikel(s)));
//aturan stemming fadillah
if (stem.equalsIgnoreCase(hapuspp(hapuspartikel(s)))) {//fail
  stem = hapusakhiran(hapusawalan2(stem));
} else {//success
String cek=stem;
stem=hapusakhiran(stem);
if (stem.equalsIgnoreCase(cek)) {//fail
stem = cek;
}else{//success
stem=hapusawalan2(stem);}}}
   }catch(Exception e){
   stem=s;
   return stem;}
return stem;}
```

Kode program tersebut merupakan implementasi dari metode *fadillah stemmer.* Urutan aturan penghapusan imbuhan disesuaikan dengan urutan metode *fadillah* yang telah dijelaskan pada sub-bab 2 dan 3.

Lampiran 5
Kode Program Metode *Asian Stemmer*

Metode *stemmer asian* merupakan pengembangan dari metode *nazief* dimana terdapat tahap pengecekan *rule precedence* dan pengecekkan kata jamak. Metode ini juga menggunakan kamus kata dasar sebagai kontrol kesalahan *understemming* dan *overstemming.* Pada metode *asian* terdapat kesamaan kode program pada method-methodnya dengan metode *nazief stemmer.* Oleh sebab itu, pada bagian ini hanya akan dijelaskan perbedaan kode program yang ada.

Method HapusPrefiksAsian ( )

```
public String HapusPrefiksAsian(String kata){//stemmer asian
String temp=kata;
DBConnect con = new DBConnect();
try{
else
if(kata.startsWith("ter")&&con.cekKonsonan(kata.charAt(3))==tru
e&&kata.charAt(3)!='r'&&kata.substring(4,
6).equals("er")&&con.cekKonsonan(kata.charAt(6))==true){//12 -
>tambahan
kata=kata.substring(3, kata.length());}
//me
else if(kata.startsWith("mempe")){//21 -> modified
kata=kata.substring(3, kata.length());}
else
if(kata.startsWith("meng")&&con.ghqk(kata.charAt(4))==true){//1
7 ->modified
kata=kata.substring(4, kata.length());}
else
if(kata.startsWith("pe")&&con.cekKonsonan(kata.charAt(2))==true
&&con.rwylmn(kata.charAt(2))==false&&kata.substring(3,
5).equals("er")&&con.cekKonsonan(kata.charAt(5))==true){//39 -
>tambahan
kata=kata.substring(2, kata.length());}
}catch(Exception e){
return temp;}
return kata;}
```

Method *HapusPrefiksAsian* sama seperti method *HapusPrefiksNazief*, namun terdapat beberapa tambahan dan modifikasi kode program. Tambahan yang ada yaitu untuk aturan penghapusan *ter* dan *pe.* Sementara modifikasi dilakukan pada aturan penghapusan *mempe* dan *meng.*

Method cekRulePrecedence ( )

```
private boolean cekRulePrecedence(String kata){
boolean rule=false;
if(kata.startsWith("be")&&kata.endsWith("lah")){
rule=true;}
if(kata.startsWith("be")&&kata.endsWith("an")){
rule=true;}
if(kata.startsWith("me")&&kata.endsWith("i")){
rule=true;}
```

Method cekRulePrecedence ( ) Lanjutan

```
if(kata.startsWith("di")&&kata.endsWith("i")){
rule=true;}
if(kata.startsWith("pe")&&kata.endsWith("i")){
rule=true;}
if(kata.startsWith("te")&&kata.endsWith("i")){
rule=true;}
return rule;}
```

Method *cekRulePrecedence* digunakan untuk mengecek kombinasi imbuhan. Apabila keluaran method bernilai *true*, maka penghapusan imbuhan dimulai dengan penghapusan prefiks terlebih dahulu.

Method cekKataJamak ( )

```
private boolean cekKataJamak(String kata){
boolean jamak=false;
int min=0;
if(kata.contains("-")&&!kata.startsWith("-")&&!kata.endsWith("-
")){
for(int i=0;i<kata.length();i++){
if(kata.charAt(i)=='-'){
min++;}}
if(min==1){
jamak=true;}}
return jamak;}
```

Method *cekKataJamak* digunakan untuk mengecek apakah kata yang dimasukkan merupakan kata jamak. Untuk mengetahuinya, dilakukan pengecekkan simbol penghubung "-" didalam kata. Method tersebut akan memberikan nilai *true* jika pada kata ditemukan hanya 1 simbol "-".

Method pisahKataJamak ( )

```
private String [] pisahKataJamak(String kata){
    String kata1="";
    String kata2="";
    String []katajamak=new String[2];
    boolean min =false;
for(int i=0;i<kata.length();i++){
if(kata.charAt(i)!='-'){
kata1=kata1+(kata.charAt(i));}
else if(kata.charAt(i)=='-'){
min=true;}
if(min==true){
kata2=kata.substring(i+1, kata.length());
break;}}
    katajamak[0]=kata1;
    katajamak[1]=kata2;
 return katajamak;}
```

Method *pisahKataJamak* dijalankan apabila hasil dari method *cekKataJamak* adalah *true.* Method ini berfungsi memisah kata sebelum simbol "-" dan setelahnya. Hasil pemisahan akan dimasukkan kedalam *array* yang berisi kedua kata tersebut.

Kode Program Implementasi Metode Asian *Stemmer*

```
public String stemmerAsian(String kata){
AsianStemmer con = new AsianStemmer();
String katadasar;
String temp=kata;
if(con.cekKataJamak(kata)==true){//katajamak
String []katajamak=con.pisahKataJamak(kata);
String kata1=katajamak[0];
String kata2=katajamak[1];
String a; String b;
if(kata1.equals(kata2)){//jika sama proses salah satu kata
saja
if(con.cekRulePrecedence(kata1)==true){
katadasar=con.stemRulePrecedence(kata1);}
else{
katadasar=con.stemNonRulePrecedence(kata1);}}
else{//jika 2 kata tidak sama, lakukan stemming untuk masing2
kata
if(con.cekRulePrecedence(kata1)==true){
a=con.stemRulePrecedence(kata1);}
else{
a=con.stemNonRulePrecedence(kata1);}
if(con.cekRulePrecedence(kata2)==true){
b=con.stemRulePrecedence(kata2);}
else{
b=con.stemNonRulePrecedence(kata2);}
if(a.equals(b)){//jika hasil stem kedua kata jamak sama, ambil
1 saja
katadasar=a;}
else{//jika hasil stem tidak sama, keduanya menjadi hasil stem
katadasar=a+" "+b;}}}
else{//jika bukan kata jamak
if(con.cekRulePrecedence(kata)==true){
katadasar=con.stemRulePrecedence(kata);}
else{
katadasar=con.stemNonRulePrecedence(kata);}}
return katadasar;}
```

Kode program tersebut digunakan untuk melakukan *stemming* dengan metode *asian stemmer.* Proses dimulai dengan pengecekkan kata jamak, kemudian pemisahan kata jamak apabila hasil pengecekkan bernilai *true.* Selanjutnya dilakukan pengecekkan *rule precedence.* Apabila bukan termasuk kata dengan *rule precedence,* maka kata akan diproses seperti metode *nazief.* Tetapi jika kata memenuhi aturan *rule precedence,* maka penghapusan prefiks akan dilakukan lebih dulu sebelum penghapusan sufiks.

Lampiran 6
Kode Program Metode *Enhanced Confix Stripping Stemmer (ECS)*

Metode ECS *stemmer* merupakan pengembangan dari metode *asian stemmer.* Pada metode ini terdapat modifikasi beberapa aturan pada penghapusan prefiks, selain itu terdapat proses proses yang disebut pengembalian akhiran. Proses pengembalian akhiran merupakan pengembangan dari proses kombinasi awalan-akhiran pada metode *arifin stemmer.* Tujuan dari proses pengembalian akhiran hampir sama dengan pengecekkan *rule precedence,* yaitu untuk mencegah apabila ada kata dasar yang sebagian hurufnya terhapus pada saat penghapusan sufiks. Hal tersebut sebagai antisipasi apabila ada aturan yang belum didefinisikan pada *rule precedence.*

Berikut ini akan dijelaskan kode program hasil pengembangan dari metode *asian stemmer* yang diterapkan pada metode ECS.

Method HapusPrefiksECS ( )

```
public String HapusPrefiksECS (String kata){//ECstemmer
DBConnect con = new DBConnect();
String temp=kata;
try{
else if(kata.startsWith("menge")){//21->tambahan
kata=kata.substring(5, kata.length());}
else if(kata.startsWith("mem")&&kata.charAt(3)=='p'){//21 -
>tambahan
kata=kata.substring(3, kata.length());}
else if(kata.startsWith("men")&&kata.charAt(3)=='s'){//19-
>tambahan
kata=kata.substring(3, kata.length());}
else if(kata.startsWith("penge")){//->tambahan
kata=kata.substring(5, kata.length());}
else if(kata.startsWith("peng")&&kata.charAt(4)=='k'){//-
>tambahan
kata=kata.substring(4, kata.length());}
}catch(Exception e){
return temp;}
return kata;}
```

Untuk penghapusan awalan, kode program yang digunakan sama dengan metode *asian* dengan penambahan beberapa aturan. Aturan yang ditambahkan adalah *menge, mem, men, penge,* dan *peng.*

Method cekPrefiksECS ( )

```
public String cekPrefiksECS(String kata){
String prefik="";
DBConnect con = new DBConnect();
try{
else if(kata.startsWith("menge")){//21->tambahan
prefik="menge";}
else if(kata.startsWith("mem")&&kata.charAt(3)=='p'){//21 -
>tambahan
prefik="mem";}
```

Method cekPrefiksECS ( ) Lanjutan

```
else if(kata.startsWith("men")&&kata.charAt(3)=='s'){//19-
>tambahan
prefik="men";}
else if(kata.startsWith("penge")){//->tambahan
prefik="penge";}
else if(kata.startsWith("peng")&&kata.charAt(4)=='k'){//-
>tambahan
prefik="peng";}
}catch(Exception e){
return prefik;}
return prefik;}
```

Untuk proses pengecekkan awalan juga ditambahkan beberapa aturan karena menyesuaikan dengan method *HapusPrefiksECS.*

Kode Program Tahap Pengembalian Akhiran (*loopPengembalianAkhiran*)

```
else{//proses kembalikan akhiran
String preremoveprefix=temp2;//kembalikan kata sebelum
penghapusan prefix

 preremoveprefix=preremoveprefix+DS;//kembalikan derivational
sufik
 preremoveprefix=con.HapusPrefiksECS(preremoveprefix);
if(con.cekKataDasar(con.connect(), preremoveprefix)==true){
katadasar=preremoveprefix;
stemstop=true;
}else{
preremoveprefix=temp2;
preremoveprefix=preremoveprefix+DS+PP;//kembalikan posesive
pronoun
preremoveprefix=con.HapusPrefiksECS(preremoveprefix);
if(con.cekKataDasar(con.connect(), preremoveprefix)==true){
katadasar=preremoveprefix;
stemstop=true;
}else{
preremoveprefix=temp2;
preremoveprefix=preremoveprefix+DS+PP+IP;//kembalikan inflek
partikel
preremoveprefix=con.HapusPrefiksECS(preremoveprefix);
if(con.cekKataDasar(con.connect(), preremoveprefix)==true){
katadasar=preremoveprefix;
stemstop=true;
}//akir proses pengembalian akiran
else{
preremoveprefix=preremoveprefix+DS+PP+IP;//recoding
preremoveprefix=con.HapusPrefiksECS(preremoveprefix);
preremoveprefix=con.PrefiksRecoding(preremoveprefix);
if(con.cekKataDasar(con.connect(), preremoveprefix)==true){
katadasar=preremoveprefix;
stemstop=true;
}
else{//seluruh proses gagal
katadasar=temp;
stemstop=true;}}}}}
```

Tahap pengembaliang akhiran dilakukan dengan tiga langkah yaitu pengembalian *derivational suffix (DS)*, pengembalian *possesive pronoun (PP)*, dan pengembalian *inflectional particle (IP)*. Pada setiap langkah pengembalian akhiran, coba lakukan penghapusan prefiks dan cek kamus kata dasar. Apabila tidak ditemukan, maka kembalikan awalan yang dihapus tersebut, dan lanjutkan dengan tahap pengembalian akhiran selanjutnya. Jika proses telah sampai pada tahap pengembalian *inflectional particle*, lakukan *recoding*. Namun jika kata dasar belum ditemukan, maka seluruh proses dianggap gagal dan masukkan awal dianggap kata dasar.

Selain kode program yang telah disebutkan, kode program untuk fungsi lainya sama dengan kode yang digunakan pada metode *asian stemmer*.

Lampiran 7
Kode Program Metode *Arifiyanti Stemmer*

Metode *stemmer arifiyanti* merupakan pengembangan dari metode *asian* yang disesuaikan dengan data *twitter*. Pada metode ini dilakukan beberapa penambahan pada aturan yang ada pada metode *asian stemmer*.

Method cekRulePrecedence ( )

```
private boolean cekRulePrecedence(String kata){
boolean rule=false;
if(kata.startsWith("di")&&kata.endsWith("in")){
rule=true;}
if(kata.startsWith("se")&&kata.endsWith("i")){
rule=true;}
if(kata.startsWith("n")&&kata.endsWith("i")){
rule=true;}
if(kata.startsWith("n")&&kata.endsWith("a")){
rule=true;}
if(kata.startsWith("te")&&kata.endsWith("k")){
rule=true;}
return rule;}
```

Method *cekRulePrecedene* pada metode *arifiyanti* sama dengan method yang digunakan pada metode *asian stemmer* namun dengan beberapa tambahan. Pada kode diatas diberikan beberapa aturan tambahan yaitu *di-in, se-i, n-i, n-a,* dan *te-k.*

Method HapusInflectionalParticle ( )

```
public String HapusInflectionalParticle(String kata){
else if(kata.endsWith("kh")){//->tambahan
kata=kata.substring(0, kata.length()-2);}
else if(kata.endsWith("pn")){//->tambahan
kata=kata.substring(0, kata.length()-2);}
return kata;}
```

Pada method *HapusInflectionalParticle* ditambahkan aturan penghapusan untuk *kh* dan *pn.*

Method HapusPosessivePronoun ( )

```
public String HapusPossesivePronoun(String kata){
else if(kata.endsWith("ny")){//->tambahan
kata=kata.substring(0, kata.length()-2);}
return kata;}
```

Pada method *HapusPosessivePronoun* ditambahkan aturan untuk penghapusan *ny.*

Method HapusDerivationalSuffix ( )

```
public String HapusDerivationalSuffix(String kata){//hapus i
,kan, an
else if(kata.endsWith("in")){//->tambahan
kata=kata.substring(0,kata.length()-2);}
return kata;}
```

Pada method *HapusDerivationalSuffix* ditambahkan aturan untuk penghapusan *in.*

Kode Program Penghapusan Huruf *r*

```
if(kata.charAt(0)=='r'){//jika awalan r, coba hapus r dan cek
kamus
kata=kata.substring(1, kata.length());
if(con.cekKataDasar(con.connect(),
kata)==true||anj.cekKataTdkBaku(anj.connect(), kata)==true){
katadasar=kata;
stemstop=true;}
else{//jika tdk ditemukan kembalikan r
kata='r'+kata;}}
```

Kode program tersebut digunakan untuk melakukan pengecekan bila suatu kata diawali huruf *r* maka coba hapus *r* dan cek kamus kata dasar. Jika ditemukan, proses dihentikan, namun jika tidak, huruf *r* dikembalikan pada kata.

Aturan tersebut merupakan modifikasi yang dilakukan pada metode *arifiyanti stemmer.*

Selain kode-kode tersebut diatas, kode program yang lain misalnya untuk penghapusan awalan dan pengecekan awalan sama dengan kode program pada metode *asian stemmer.*

Lampiran 8
Kode R Untuk Pengambilan Data Twitter

Pada lampiran ini akan diberikan kode program *r* yang digunakan untuk pengambilan data dari *twitter.*

Kode R Untuk Pengambilan Data Twitter

```
> install.packages(c("twitteR", "httr", "DBI", "bit32",
"rjson", "httk", "httpuv", "base32enc"))
#tunggu proses instalasi
> library(twitteR)
> library(httr)
> library(DBI)
> library(bit64)
> library(rjson)
> library(httpuv)
> library(httk)
#jalankan library
> api_key = "dbITbfws5zISJCgY3kv0krXQA"
> api_secret =
"vIyOLBZAacttTZTN3A1Fkra8Lo7ERue73SyVB5cBnoNNGKC0Zu"
> access_token = "2350917865-
8iFYnprIXBaEJJA4hP7tKcmfGDorLhKX8k2a1b9"
> access_token_secret =
"jAkG9C6HjfcQv0MMXqRajg3l3dw8qMHFTRNgm3O11RIh8"
> setup_twitter_oauth(api_key, api_secret, access_token,
access_token_secret)
[1] "Using direct authentication"
Use a local file to cache OAuth access credentials between R
sessions?
1: Yes
2: No
Selection: 1
#masukkan api_key, api_secret, access_token, dan
access_token_secret
> tl.asap = searchTwitter("@pln_123", n=10000)
> text = sapply(tl.asap, statusText)
#proses pengambilan data
> asap.df = twListToDF(tl.asap)
> write.csv(asap.df, file="DataPLN.csv")
#proses penyimpanan data
```

Tahap pertama yang perlu dilakukan adalah melakukan proses instalasi *library* yang dibutuhkan sebab fungsi pengambilan data *twiter* tidak tersedia pada *r*. Apabila sudah pernah melakukan instalasi, dapat langsung melakukan pemanggilan *library.*

Tahap selanjutnya adalah mengakses *twitter* dengan *api_key*, *api_secret*, *access_token*, dan *access_token_secret*yang telah didapatkan. Sebagai catatan, kita dapat menggunakan *key* dan *token* milik orang lain apabila kita tidak ingin melakukan registrasi pada *twitter aplication management,* selama akun tersebut aktif.

Setelah tahap tersebut, pencarian dan pengumpulan data dapat dilakukan, namun data yang dikumpulkan terbatas hanya sembilan hari yang lampau. Untuk itu, pengambilan data dapat dilakukan setiap sembilan hari sekali.

Lampiran 9
Kode R Untuk *Clustering* Data Dengan Metode *K-means* dan Evaluasi Menggunakan DBI

Pada bagian ini akan dijelaskan langkah-langkah melakukan pengelompokan data pada *r* menggunakan metode *k-means.* Berikut ini adalah kode program *r* untuk melakukan *clustering* dan evaluasi hasil *cluster.*

Kode R Untuk *Clustering* dan Evaluasi *Cluster*

```
> install.packages('tm',repos = "https://cran.r-project.org")
#install library text mining (tm)

> library(tm)
Loading required package: NLP
Warning messages:
1: package 'tm' was built under R version 3.3.3
2: package 'NLP' was built under R version 3.3.2
> 1
[1] 1
#jalankan library tm

> setwd("C://Users//HP//Desktop//Data")
> reviews <- read.csv("HasilStem.csv",stringsAsFactors = FALSE)
> review_source <- VectorSource(reviews$data)
#masukkan data

> corpus <- Corpus(review_source)
> dtm <- DocumentTermMatrix(corpus)
> dtm2 <- as.matrix(dtm)
#transformasi data menjadi matriks

> dtm_tfidf <- weightTfIdf(dtm)
> m <- as.matrix(dtm_tfidf)
> rownames(m) <- 1:nrow(m)
#tahap seleksi term frequency invers document frequency

> norm_eucl <- function(m)
+ m/apply(m,1, function(x) sum(x^2)^.5)
> m_norm <- norm_eucl(m)
#hitung euclidean

> result <- kmeans(m_norm,4,100)
#clustering k-means

> install.packages('clusterSim',repos = "https://cran.r-
project.org")
> library(clusterSim)
> dbs <- index.DB(m_norm,result$cluster,centrotypes =
"centroids")
> dbs$DB
#kode untuk evaluasi menggunakan DBI
```

Kode program untuk *clustering* data *twitter* menggunakan metode *k-means* dimulai dengan menginstal *library tm* atau *text mining.* Selanjutnya, setelah *library* diaktifkan, data dimasukkan dengan terlebih dahulu mendefinisikan lokasi *file,* kemudian masukkan nama *file.*

Sebelum dilakukan tahap *clustering,* data perlu ditransformasi menjadi matriks. Pada kode program diatas, telah disertakan keterangan tambahan mengenai fungsi untuk transformasi data beserta tahap *tf-idf.*

Untuk evaluasi menggunakan *davies bouldin index* (DBI), dibutuhkan *library clusterSim.* Nilai DBI dihitung dengan memasukkan data sebelum *clustering,* dan hasil *cluster.*

Pada penelitian ini, parameter *clustering k-means* untuk seluruh kelas dan seluruh metode *stemmer* dibiarkan sama, serta standar. Hal ini dilakukan agar hasil perbandingan metode *stemmer* valid.

Lampiran 10
Kode R Untuk Pengujian Statistik *t*

Pada bagian ini akan dijelaskan langkah-langkah untuk melakukan pengujian *t* dengan menggunakan *r.* Berikut ini adalah kode program untuk melakukan uji *t.*

Kode R Untuk Pengujian Statistik *t*

```
> setwd("C://Users//HP//Desktop//Data")
> reviews <- read.csv("HasilDBI.csv",stringsAsFactors = FALSE)
> attach(reviews)
> names(reviews)
 [1] "Kelas"                    "TanpaStemmer"
 [3] "Arifiyanti"               "Enhanced.Confix.Stripping"
 [5] "Asian"                    "Fadillah"
 [7] "Arifin"                   "Nazief"
 [9] "Porter.confix.stripping"  "tes3"
> t.test(Fadillah,TanpaStemmer,mu=0,alt="two.sided",paired =
T,conf.level = 0.95)
```

Sebelum melakukan uji *t* data pengujian perlu dipersiapkan. Data yang berupa kumpulan nilai DBI dikumpulkan berdasarkan metode *stemmer*nya dan disimpan dalam format *csv.* Data dimasukkan dengan terlebih dahulu mendefinisikan lokasi *file,* kemudian masukkan nama *file.*

Kode tersebut merupakan contoh pengujian untuk nilai DBI yang didapat metde *fadillah* dengan tanpa penerapan *stemmer.* Pengujian dilakukan dengan metoda *paired t test* dengan α=0,95.

Lampiran 11
Hasil Pengujian Korelasi Pearson Antara Performa *Stemmer* Dan Performa *Clustering*

```
> setwd("C://Users//HP//Desktop//Data")
> reviews <- read.csv("UjiPearson.csv",stringsAsFactors = FALSE)
> attach(reviews)
> names(reviews)
[1] "nomor"     "understem" "overstem"  "DBI"

> cor.test(understem,DBI, method = "pearson")
        Pearson's product-moment correlation
data:  understem and DBI
t = -4.6419, df = 5, p-value = 0.005623
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9854227 -0.4598310
sample estimates:
      cor
-0.9009193

> cor.test(overstem,DBI, method = "pearson")
        Pearson's product-moment correlation
data:  overstem and DBI
t = 5.9189, df = 5, p-value = 0.001962
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6172215 0.9906509
sample estimates:
     cor
0.9354698
```

Lampiran 12
Hasil Perhitungan *P-Values* Pada Pengujian Signifikansi *Paired t Test*

```
> names(reviews)
[1] "Kelas"                  "TanpaStemmer"
[3] "Arifiyanti"             "Enhanced.Confix.Stripping"
[5] "Asian"                  "Fadillah"
[7] "Arifin"                 "Nazief"
[9] "Porter.confix.stripping"
> dim(reviews)
[1] 19  9
>
boxplot(TanpaStemmer,Porter.confix.stripping,Nazief,Arifin,Fadillah,Asian,Enhanced.Confix.Stri
pping,Arifiyanti)
> # H0: Mean differences in reviews is 0
> #two-sided test

> t.test(TanpaStemmer,Arifiyanti,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-tes
data:  TanpaStemmer and Arifiyanti
t = 3.3466, df = 18, p-value = 0.003592
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1666548 0.7288188
sample estimates:
mean of the differences
        0.4477368

> t.test(TanpaStemmer,Enhanced.Confix.Stripping,mu=0,alt="two.sided", paired = T, conf.level =
0.95)
        Paired t-test
data:  TanpaStemmer and Enhanced.Confix.Stripping
t = 1.5385, df = 18, p-value = 0.1413
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.08618527  0.55765895
sample estimates:
mean of the differences
        0.2357368

> t.test(TanpaStemmer,Asian,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  TanpaStemmer and Asian
t = 4.6919, df = 18, p-value = 0.0001817
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1905455 0.4995598
sample estimates:
mean of the differences
        0.3450526

> t.test(TanpaStemmer,Fadillah,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  TanpaStemmer and Fadillah
t = 0.62829, df = 18, p-value = 0.5377
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
 -0.2203222  0.4083222
sample estimates:
mean of the differences
               0.094

> t.test(TanpaStemmer,Arifin,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  TanpaStemmer and Arifin
t = 1.5117, df = 18, p-value = 0.148
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.07621504  0.46726767
sample estimates:
mean of the differences
            0.1955263

> t.test(TanpaStemmer,Nazief,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  TanpaStemmer and Nazief
t = 2.7197, df = 18, p-value = 0.01405
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.0641482 0.4997465
sample estimates:
mean of the differences
            0.2819474

> t.test(TanpaStemmer,Porter.confix.stripping,mu=0,alt="two.sided", paired = T, conf.level =
0.95)
            Paired t-test
data:  TanpaStemmer and Porter.confix.stripping
t = -1.2435, df = 18, p-value = 0.2296
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4652846  0.1192846
sample estimates:
mean of the differences
              -0.173

> t.test(Arifiyanti,TanpaStemmer,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and TanpaStemmer
t = -2.6902, df = 18, p-value = 0.01496
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.65585621 -0.08067011
sample estimates:
mean of the differences
           -0.3682632

> t.test(Arifiyanti,Asian,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Asian
t = -0.19672, df = 18, p-value = 0.8463
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
```

```
 -0.2710989  0.2246779
sample estimates:
mean of the differences
          -0.02321053

> t.test(Arifiyanti,Arifin,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Arifin
t = -1.4881, df = 18, p-value = 0.154
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.41660420  0.07113051
sample estimates:
mean of the differences
           -0.1727368

> t.test(Arifiyanti,Nazief,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Nazief
t = -0.72402, df = 18, p-value = 0.4784
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3367831  0.1641516
sample estimates:
mean of the differences
          -0.08631579

> t.test(Arifiyanti,Fadillah,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Fadillah
t = -1.4219, df = 18, p-value = 0.1721
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6794895  0.1309632
sample estimates:
mean of the differences
           -0.2742632

> t.test(Arifiyanti,Enhanced.Confix.Stripping,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Enhanced.Confix.Stripping
t = -0.75505, df = 18, p-value = 0.46
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5012797  0.2362271
sample estimates:
mean of the differences
           -0.1325263

> t.test(Arifiyanti,Porter.confix.stripping,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti and Porter.confix.stripping
t = -3.0523, df = 18, p-value = 0.006858
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9138154 -0.1687109
sample estimates:
```

```
mean of the differences
          -0.5412632

> t.test(Enhanced.Confix.Stripping,Asian,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
           Paired t-test
data:  Enhanced.Confix.Stripping and Asian
t = 0.61621, df = 18, p-value = 0.5455
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2633884  0.4820200
sample estimates:
mean of the differences
           0.1093158

> t.test(Enhanced.Confix.Stripping,Fadillah,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
           Paired t-test
data:  Enhanced.Confix.Stripping and Fadillah
t = -0.83071, df = 18, p-value = 0.417
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5001995  0.2167258
sample estimates:
mean of the differences
          -0.1417368

> t.test(Enhanced.Confix.Stripping,Arifin,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
           Paired t-test
data:  Enhanced.Confix.Stripping and Arifin
t = -0.20376, df = 18, p-value = 0.8408
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4548030  0.3743819
sample estimates:
mean of the differences
         -0.04021053

> t.test(Enhanced.Confix.Stripping,Nazief,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
           Paired t-test
data:  Enhanced.Confix.Stripping and Nazief
t = 0.30103, df = 18, p-value = 0.7668
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2762944  0.3687154
sample estimates:
mean of the differences
          0.04621053

> t.test(Enhanced.Confix.Stripping,Porter.confix.stripping,mu=0,alt="two.sided", paired = T,
conf.level = 0.95)
           Paired t-test
data:  Enhanced.Confix.Stripping and Porter.confix.stripping
t = -3.1676, df = 18, p-value = 0.005329
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6798348 -0.1376389
sample estimates:
mean of the differences
```

```
          -0.4087368

> t.test(Asian,Fadillah,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  Asian and Fadillah
t = -1.6127, df = 18, p-value = 0.1242
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.57810139  0.07599613
sample estimates:
mean of the differences
          -0.2510526

> t.test(Asian,Arifin,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  Asian and Arifin
t = -1.4302, df = 18, p-value = 0.1698
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.36918302  0.07013039
sample estimates:
mean of the differences
          -0.1495263

> t.test(Asian,Nazief,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  Asian and Nazief
t = -0.6929, df = 18, p-value = 0.4972
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2544449  0.1282344
sample estimates:
mean of the differences
          -0.06310526

> t.test(Asian,Porter.confix.stripping,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  Asian and Porter.confix.stripping
t = -3.1981, df = 18, p-value = 0.004984
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8583813 -0.1777240
sample estimates:
mean of the differences
          -0.5180526

> t.test(Fadillah,Arifin,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
            Paired t-test
data:  Fadillah and Arifin
t = 0.52664, df = 18, p-value = 0.6049
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3034900  0.5065427
sample estimates:
mean of the differences
           0.1015263
```

```
> t.test(Fadillah,Nazief,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  Fadillah and Nazief
t = 1.1929, df = 18, p-value = 0.2484
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1430703  0.5189650
sample estimates:
mean of the differences
        0.1879474

> t.test(Fadillah,Porter.confix.stripping,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  Fadillah and Porter.confix.stripping
t = -2.326, df = 18, p-value = 0.0319
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.50816274 -0.02583726
sample estimates:
mean of the differences
          -0.267

> t.test(Arifin,Nazief,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  Arifin and Nazief
t = 0.90172, df = 18, p-value = 0.3791
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1149325  0.2877746
sample estimates:
mean of the differences
        0.08642105

> t.test(Arifin,Porter.confix.stripping,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  Arifin and Porter.confix.stripping
t = -1.9164, df = 18, p-value = 0.07133
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.77253352  0.03548089
sample estimates:
mean of the differences
        -0.3685263

> t.test(Nazief,Porter.confix.stripping,mu=0,alt="two.sided", paired = T, conf.level = 0.95)
        Paired t-test
data:  Nazief and Porter.confix.stripping
t = -3.2916, df = 18, p-value = 0.004056
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7453269 -0.1645679
sample estimates:
mean of the differences
        -0.4549474
```

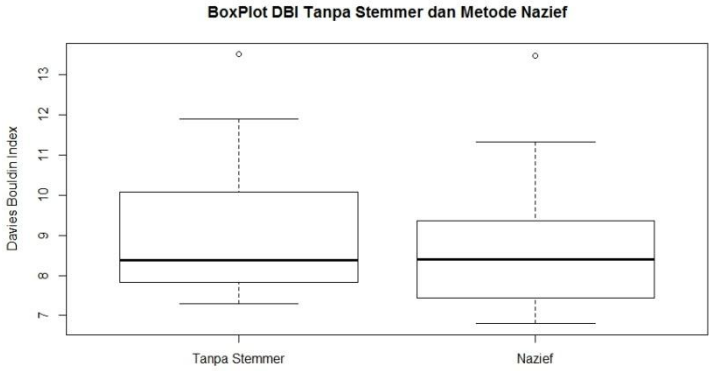
```
> setwd("C://Users//HP//Desktop//Data")
> reviews <- read.csv("HasilDBI.csv",stringsAsFactors = FALSE)
> attach(reviews)
> names(reviews)
 [1] "Kelas"                  "TanpaStemmer"
 [3] "Arifiyanti"             "Enhanced.Confix.Stripping"
 [5] "Asian"                  "Fadillah"
 [7] "Arifin"                 "Nazief"
 [9] "Porter.confix.stripping"   "Arifiyanti.aturanECS"
[11] "Arifiyanti.aturanbaruRP"

> t.test(Arifiyanti.aturanECS,TanpaStemmer,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and TanpaStemmer
t = -2.0782, df = 18, p-value = 0.05227
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.651437805  0.003543069
sample estimates:
mean of the differences
          -0.3239474

> t.test(Arifiyanti.aturanECS,Porter.confix.stripping,mu=0,alt="two.sided",paired = T,conf.level =
0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Porter.confix.stripping
t = -4.2052, df = 18, p-value = 0.0005321
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.7452253 -0.2486695
sample estimates:
mean of the differences
          -0.4969474

> t.test(Arifiyanti.aturanECS,Nazief,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Nazief
t = -0.3494, df = 18, p-value = 0.7308
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2945458  0.2105458
sample estimates:
mean of the differences
            -0.042

> t.test(Arifiyanti.aturanECS,Fadillah,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Fadillah
t = -1.5182, df = 18, p-value = 0.1463
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.54814661  0.08825187
sample estimates:
mean of the differences
          -0.2299474
```

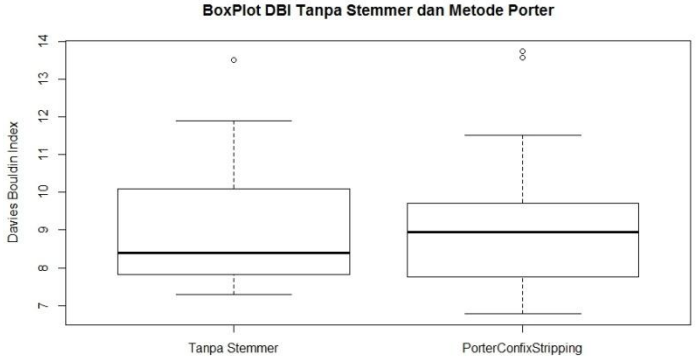
```
> t.test(Arifiyanti.aturanECS,Arifin,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Arifin
t = -0.70356, df = 18, p-value = 0.4907
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5119026  0.2550605
sample estimates:
mean of the differences
        -0.1284211

> t.test(Arifiyanti.aturanECS,Asian,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
Paired t-test
data:  Arifiyanti.aturanECS and Asian
t = 0.1264, df = 18, p-value = 0.9008
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3296951  0.3719056
sample estimates:
mean of the differences
        0.02110526

> t.test(Arifiyanti.aturanECS,Arifiyanti,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Arifiyanti
t = 0.26768, df = 18, p-value = 0.792
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3035041  0.3921357
sample estimates:
mean of the differences
        0.04431579

> t.test(Arifiyanti.aturanECS,Arifiyanti.aturanbaruRP,mu=0,alt="two.sided",paired = T,conf.level
= 0.95)
        Paired t-test
data:  Arifiyanti.aturanECS and Arifiyanti.aturanbaruRP
t = 0.73946, df = 18, p-value = 0.4692
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1816947  0.3790631
sample estimates:
mean of the differences
        0.09868421

t.test(Arifiyanti.aturanbaruRP,TanpaStemmer,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
        Paired t-test
data:  Arifiyanti.aturanbaruRP and TanpaStemmer
t = -5.1418, df = 18, p-value = 6.838e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5953177 -0.2499455
sample estimates:
mean of the differences
        -0.4226316
```

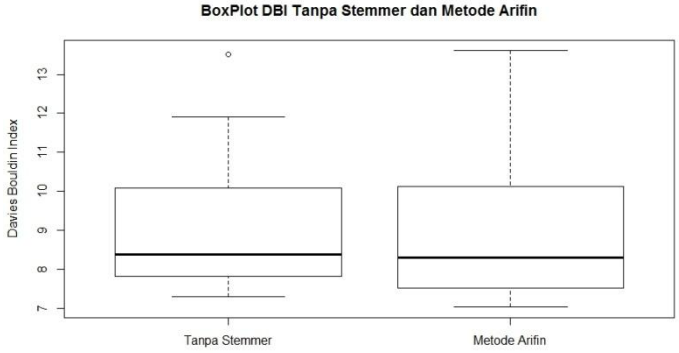
```
>  t.test(Arifiyanti.aturanbaruRP,Porter.confix.stripping,mu=0,alt="two.sided",paired =
T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti.aturanbaruRP and Porter.confix.stripping
t = -4.0784, df = 18, p-value = 0.0007054
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9024631 -0.2888000
sample estimates:
mean of the differences
          -0.5956316

>  t.test(Arifiyanti.aturanbaruRP,Nazief,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti.aturanbaruRP and Nazief
t = -1.7279, df = 18, p-value = 0.1011
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.31174385  0.03037543
sample estimates:
mean of the differences
          -0.1406842

>  t.test(Arifiyanti.aturanbaruRP,Arifin,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti.aturanbaruRP and Arifin
t = -1.9963, df = 18, p-value = 0.06125
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.46611108  0.01190055
sample estimates:
mean of the differences
          -0.227105

>  t.test(Arifiyanti.aturanbaruRP,Fadillah,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti.aturanbaruRP and Fadillah
t = -2.4417, df = 18, p-value = 0.02517
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.61140281 -0.04586035
sample estimates:
mean of the differences
          -0.3286316

>  t.test(Arifiyanti.aturanbaruRP,Asian,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
            Paired t-test
data:  Arifiyanti.aturanbaruRP and Asian
t = -0.95807, df = 18, p-value = 0.3507
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2476992  0.0925413
sample estimates:
mean of the differences
          -0.07757895
```

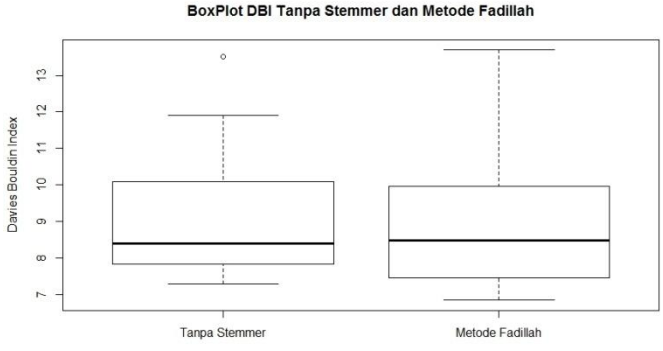
```
>  t.test(Arifiyanti.aturanbaruRP,Arifiyanti,mu=0,alt="two.sided",paired = T,conf.level = 0.95)
Paired t-test
data:  Arifiyanti.aturanbaruRP and Arifiyanti
t = -0.46004, df = 18, p-value = 0.651
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3026595  0.1939226
sample estimates:
mean of the differences
          -0.05436842

>  t.test(Arifiyanti.aturanbaruRP,Arifiyanti.aturanECS,mu=0,alt="two.sided",paired = T,conf.level
= 0.95)
              Paired t-test
data:  Arifiyanti.aturanbaruRP and Arifiyanti.aturanECS
t = -0.73946, df = 18, p-value = 0.4692
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3790631  0.1816947
sample estimates:
mean of the differences
          -0.09868421
```

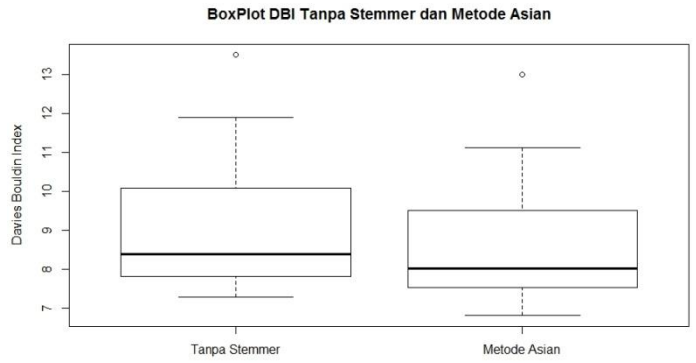
Lampiran 13 *BoxPlot* Nilai DBI Metode-Metode *Stemmer*

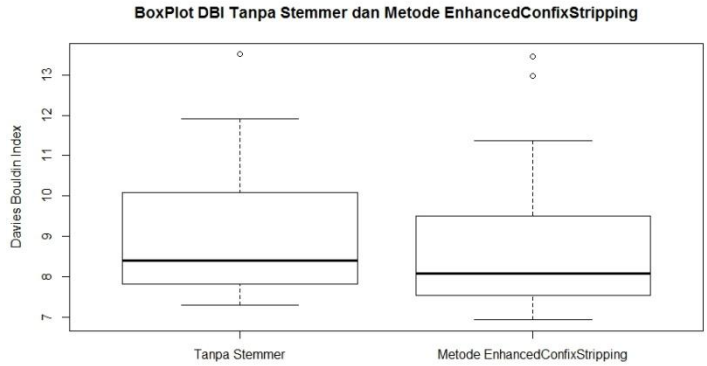


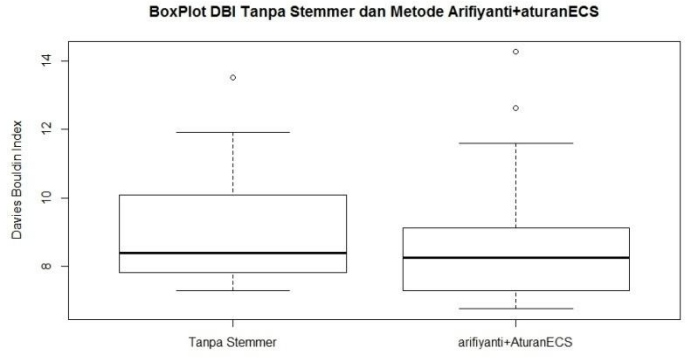


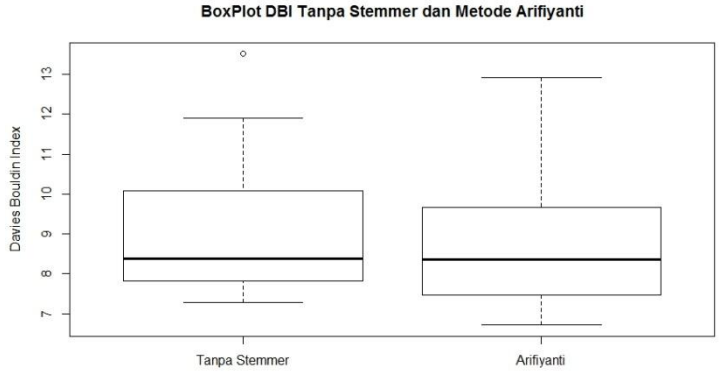




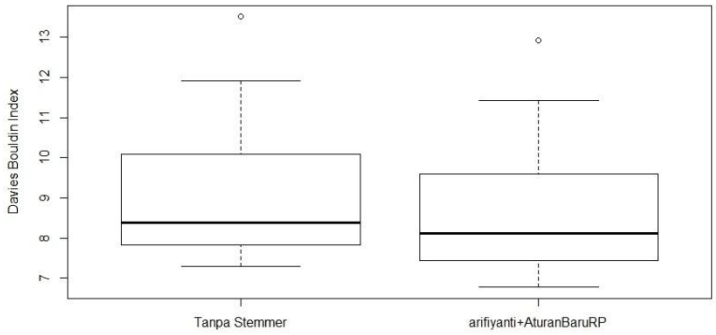

BoxPlot DBI Tanpa Stemmer dan Metode Asian
Davies Bouldin Index
7
8
9
10
11
12
13
Tanpa Stemmer
Metode Asian


BoxPlot DBI Tanpa Stemmer dan Metode EnhancedConfixStripping
Davies Bouldin Index
7
8
9
10
11
12
13
Tanpa Stemmer
Metode EnhancedConfixStripping

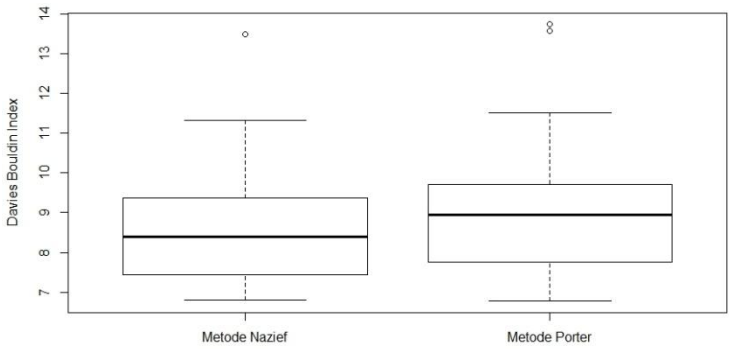

BoxPlot DBI Tanpa Stemmer dan Metode Arifiyanti+aturanECS
Davies Bouldin Index
8
10
12
14
Tanpa Stemmer
arifiyanti+AturanECS


BoxPlot DBI Tanpa Stemmer dan Metode Arifiyanti
Davies Bouldin Index
7
8
9
10
11
12
13
Tanpa Stemmer
Arifiyanti

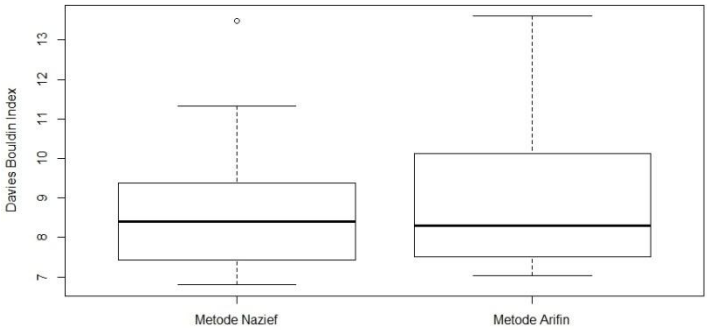
BoxPlot DBI Tanpa Stemmer dan Metode Arifiyanti+aturanbaruRP



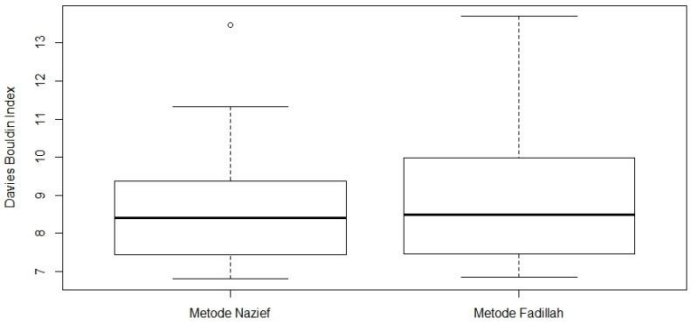
BoxPlot DBI Metode Nazief dan Metode PorterConfixStripping

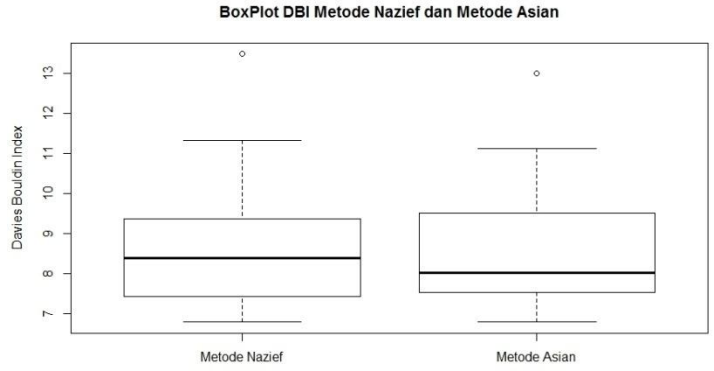


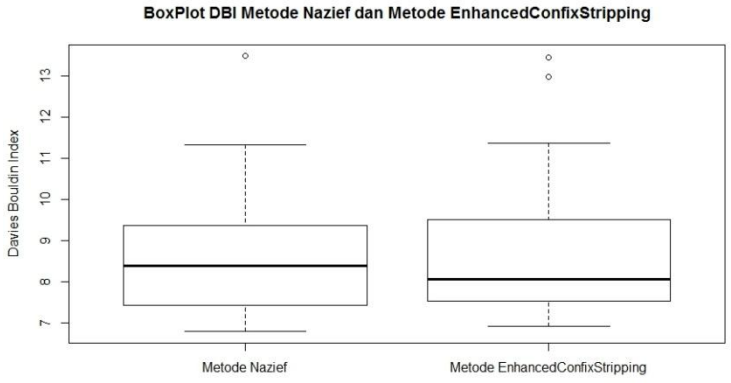
BoxPlot DBI Metode Nazief dan Metode Arifin

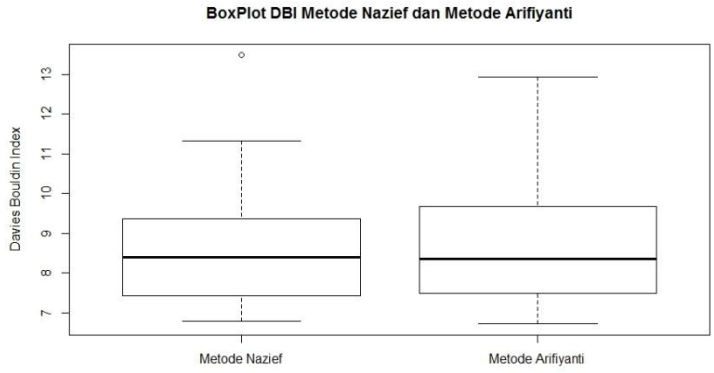


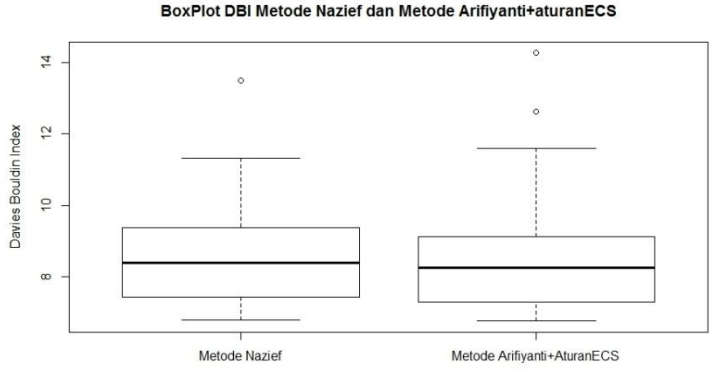
BoxPlot DBI Metode Nazief dan Metode Fadillah

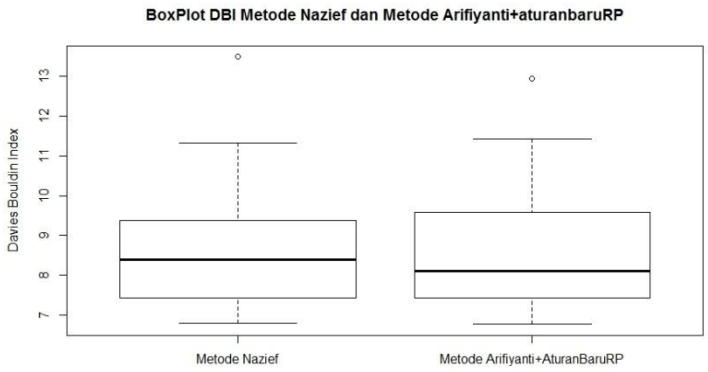


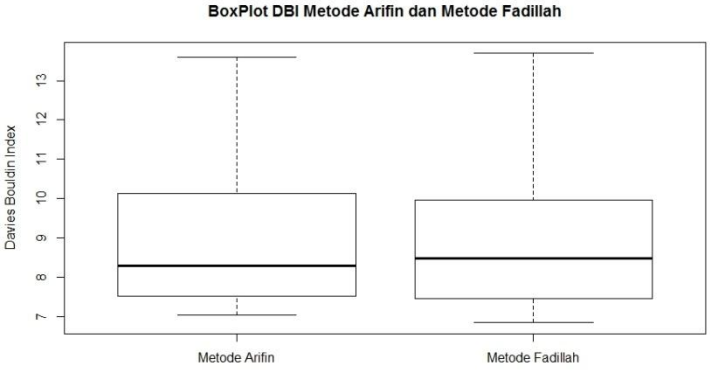

BoxPlot DBI Metode Nazief dan Metode Asian
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Nazief
Metode Asian

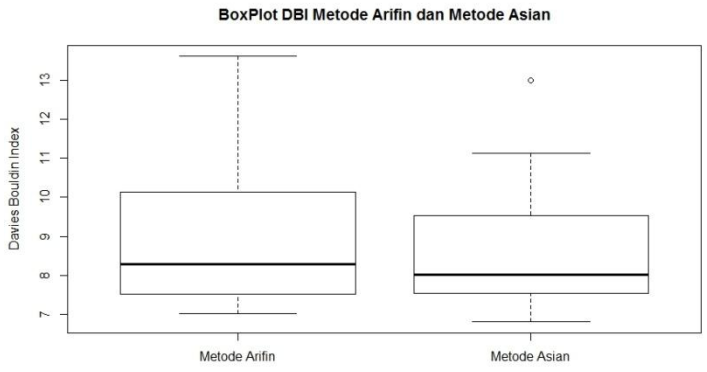

BoxPlot DBI Metode Nazief dan Metode EnhancedConfixStripping
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Nazief
Metode EnhancedConfixStripping

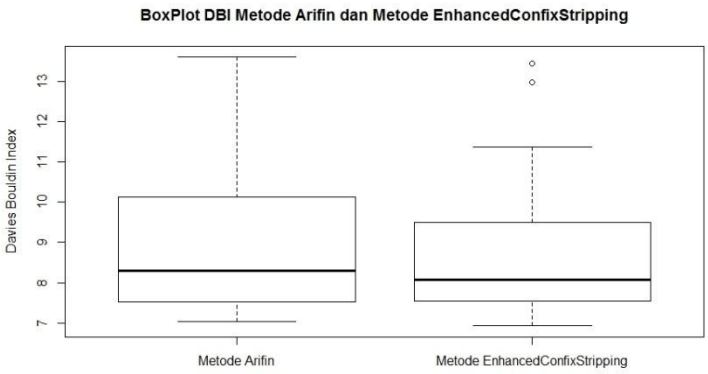

BoxPlot DBI Metode Nazief dan Metode Arifiyanti
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Nazief
Metode Arifiyanti

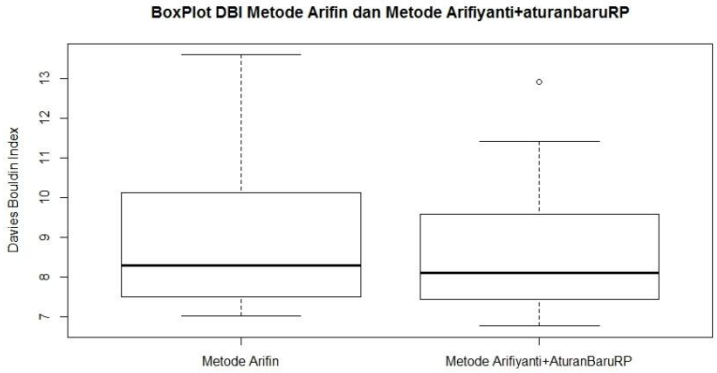

BoxPlot DBI Metode Nazief dan Metode Arifiyanti+aturanECS
Davies Bouldin Index
8
10
12
14
Metode Nazief
Metode Arifiyanti+AturanECS

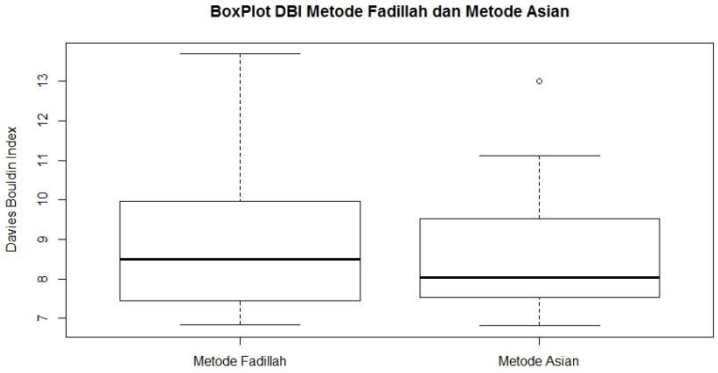

BoxPlot DBI Metode Nazief dan Metode Arifiyanti+aturanbaruRP
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Nazief
Metode Arifiyanti+AturanBaruRP

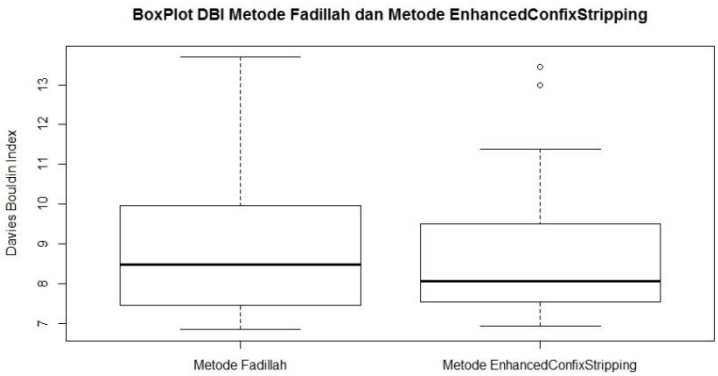

BoxPlot DBI Metode Arifin dan Metode Fadillah
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Arifin
Metode Fadillah

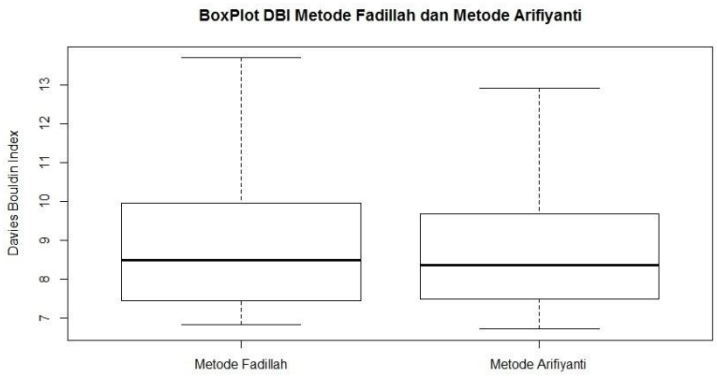

BoxPlot DBI Metode Arifin dan Metode Asian
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Arifin
Metode Asian

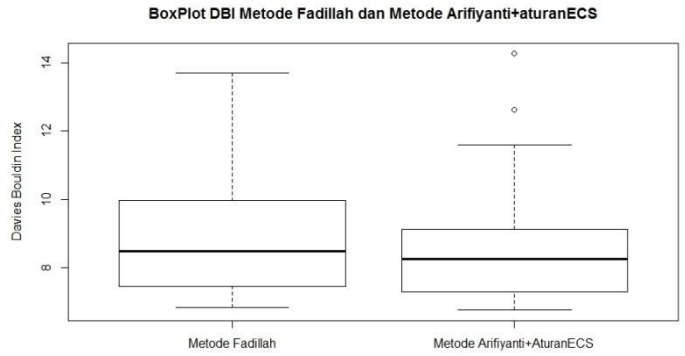

BoxPlot DBI Metode Arifin dan Metode EnhancedConfixStripping
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Arifin
Metode EnhancedConfixStripping

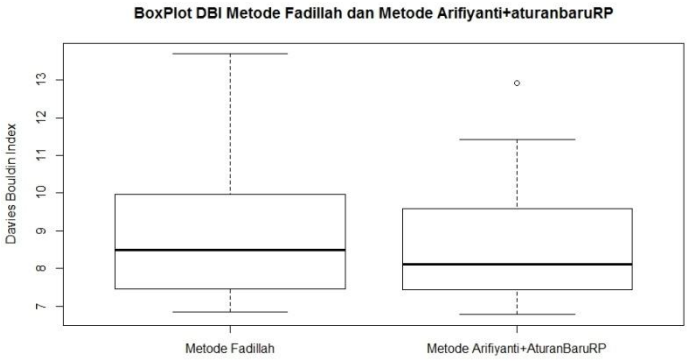

BoxPlot DBI Metode Arifin dan Metode Arifiyanti+aturanbaruRP
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Arifin
Metode Arifiyanti+AturanBaruRP

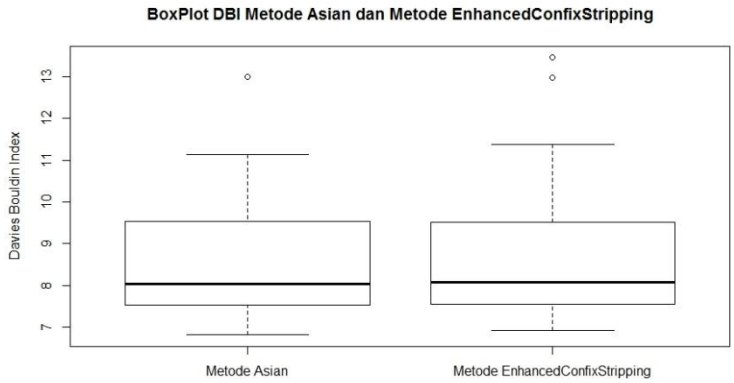

BoxPlot DBI Metode Fadillah dan Metode Asian
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Fadillah
Metode Asian

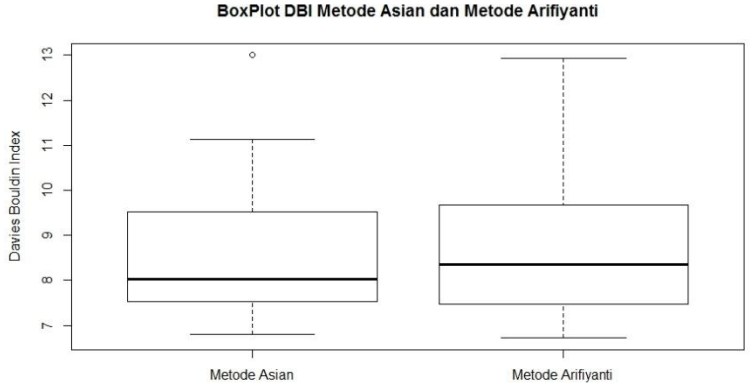

BoxPlot DBI Metode Fadillah dan Metode EnhancedConfixStripping
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Fadillah
Metode EnhancedConfixStripping


BoxPlot DBI Metode Fadillah dan Metode Arifiyanti
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Fadillah
Metode Arifiyanti

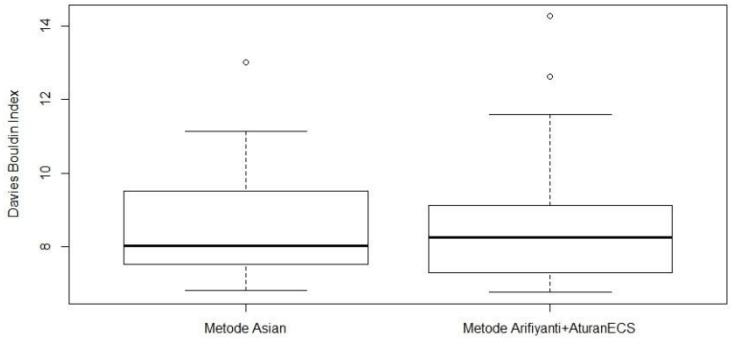

BoxPlot DBI Metode Fadillah dan Metode Arifiyanti+aturanECS
Davies Bouldin Index
8
10
12
14
Metode Fadillah
Metode Arifiyanti+AturanECS


BoxPlot DBI Metode Fadillah dan Metode Arifiyanti+aturanbaruRP
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Fadillah
Metode Arifiyanti+AturanBaruRP

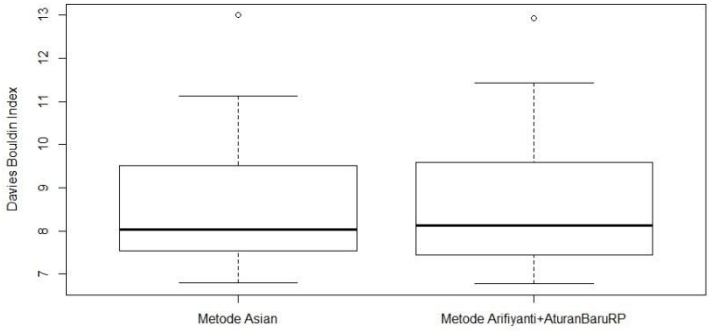

BoxPlot DBI Metode Asian dan Metode EnhancedConfixStripping
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Asian
Metode EnhancedConfixStripping


BoxPlot DBI Metode Asian dan Metode Arifiyanti
Davies Bouldin Index
7
8
9
10
11
12
13
Metode Asian
Metode Arifiyanti

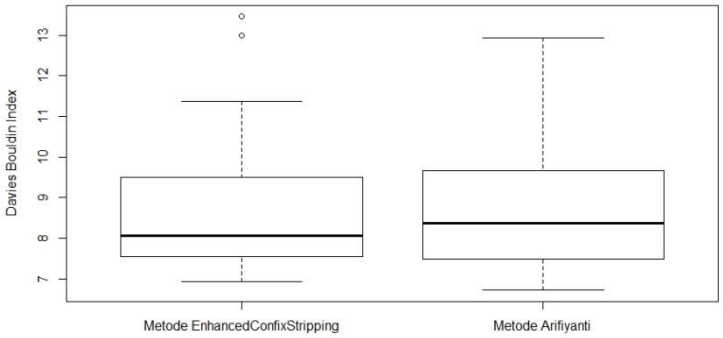
BoxPlot DBI Metode Asian dan Metode Arifiyanti+aturanECS



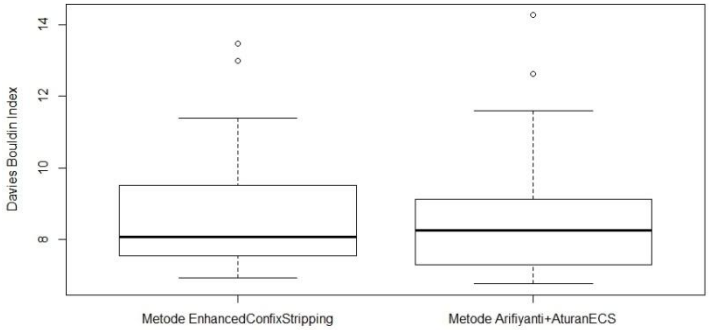
BoxPlot DBI Metode Asian dan Metode Arifiyanti+aturanbaruRP

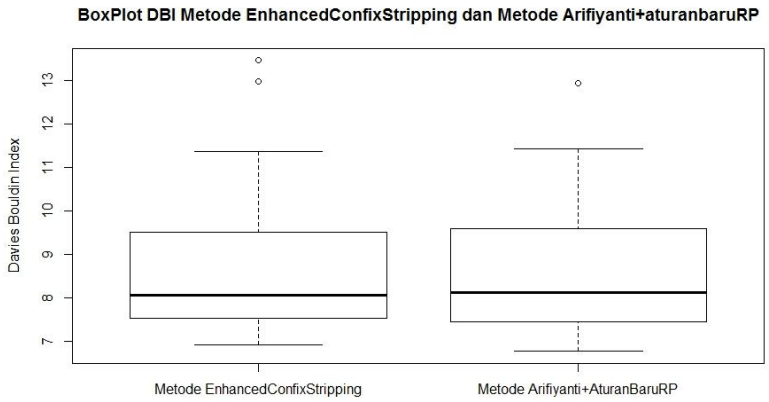


BoxPlot DBI Metode EnhancedConfixStripping dan Metode Arifiyanti



BoxPlot DBI Metode EnhancedConfixStripping dan Metode Arifiyanti+aturanECS




BoxPlot DBI Metode EnhancedConfixStripping dan Metode Arifiyanti+aturanbaruRP
Davies Bouldin Index
7
8
9
10
11
12
13
Metode EnhancedConfixStripping
Metode Arifiyanti+AturanBaruRP

BIODATA PENULIS

Afian Syafaadi Rizki. Lahir di Malang pada 16 November 1990. Merupakan anak pertama dari tiga bersaudara. Penulis menempuh pendidikan formal di SDN Lowokwaru II Malang, SMP Negeri 5 Malang, dan SMA Negeri 1 Malang. Pada tahun 2009, penulis melanjutan pendidikan Strata-1 pada Program Studi Ilmu Komputer di Universitas Brawijaya Malang. Pada tahun 2014, penulis menyelesaikan tugas akhir yang berjudul "*Text Mining*, Klasifikasi Soal Biologi Sekolah Menengah Atas Dengan Metode *Improved-KNN*". Pada tahun yang sama setelah kelulusan, penulis melanjutkan pendidikan S2 pada jurusan Sistem Informasi di Institut Teknologi Sepuluh Nopember Surabaya. Penulis dapat dihubungi melalui E-mail : afianrizki@gmail.com